196857

STIC
Scientific & Technical Information Center

STIC EIC 2100
Search Request Form

Today's Date: 7/28/06

What date would you like to use to limit the search?
Priority Date: Other:

Name Mahammad Siddiqi
AU 2154 Examiner # 79995
Room # 4C24 Phone 571-272-3976
Serial # 09/802,405

Format for Search Results (Circle One):
PAPER DISK EMAIL

Where have you searched so far?
USP DWPI EPO JPO ACM IBM TDB
IEEE INSPEC SPI Other

Is this a "Fast & Focused" Search Request? (Circle One) YES NO
A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

Is this request for a BOARD of APPEALS case? (Circle One) YES NO

Service guarantees per thread/client per
basis in a non blocking system-on-a-chip
such as DRAM environment.

STIC Searcher LANCE SEALEY Phone 2-8666
Date picked up 7/28/06 Date Completed 7/28/06

sira
Search and Information Resources Administration

STIC Search Report
EIC 2100

STIC Database Tracking Number: 196114

TO: Mohammad Siddiqui
Location: RND 4C24
Art Unit: 2154
Saturday, July 29, 2006

Case Serial Number: 09/802,405

From: Lance Sealey
Location: EIC 2100
RND-4B11
Phone: 571-272-8666

Lance.Sealey@uspto.gov

Search Notes

Dear Mohammad,

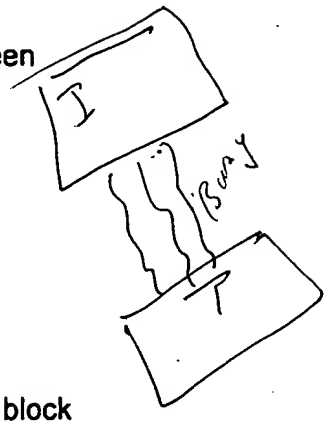
These were the closest references I could find.

Please let me know if you have any questions.

Lance

System-on-a-Chip
Multi-threaded

Claims 1, 5, 7, 9-12, 14, 15, 17-20, 24, 26, 31-32 and 34-35 are amended. Claims 37 and 38 have been added.



1. (Currently Amended) A method for communicating data between functional blocks in a computing device, comprising:

establishing a unique first~~thread~~ identifier for each independent transaction~~data~~ stream between an initiator functional block and a target functional block within a multiple threading system, wherein a plurality of independent transaction~~data~~ streams exist between the initiator functional block and the target functional block,

if the target functional block is unable to accept a data transfer from the initiator functional block, the target functional block issuing a busy signal identified by the first~~thread~~ identifier;

the initiator functional block withholding issuance of data transfers associated with the first~~thread~~ identifier in response to the issued busy signal, wherein data transfers not associated with the first~~thread~~ identifier identified by ~~the~~ issued busy signal may be issued; and

mapping a data flow from the initiator functional block to the target functional block to a ~~thread~~first transaction stream indicated by ~~the~~ its unique first~~thread~~ identifier to meet a service guarantee on a per first~~thread~~ identifier basis.

7/10/2004

2. (Original) The method as set forth in claim 1, wherein the busy signal comprises a signal that is maintained active when the target functional block is unable to accept data transfers.

3. (Original) The method as set forth in claim 1, wherein the busy signal comprises a credit signal used to communicate a number of credits that indicate how many data transfers the target functional block can accept.

4. (Original) The method as set forth in claim 3, further comprising decrementing the number of credits for each active data transfer and incrementing the number of credits upon freeing up of resources for further data transfers.

5. (Currently Amended) The method as set forth in claim 3, wherein the credit signal is generated by maintaining the credit signal in an active state for a number of clock cycles corresponding to the number of credits.

6. (Previously Presented) The method as set forth in claim 3, wherein the credit signal comprises a multi-bit coded signal indicative of the number of credits.

7. (Currently Amended) The method as set forth in claim 1, further comprising determining service guarantees for at least one transaction stream between a plurality of initiator functional blocks and ~~the~~target functional blocks.

8. (Original) The method as set forth in claim 1, further comprising the initiator functional block stopping to send data transfers so that the target functional block receives no more than a determined number of data transfers after issuance of the busy signal.

9. (Currently Amended) The method as set forth in claim 1, wherein the target functional block issues a~~the~~ busy signal no more than a determined number of clock cycles after the target functional block determines that it has insufficient buffer space to receive data transfers from ~~an~~the initiator functional block.

10. (Currently Amended) The method as set forth in claim 8, further comprising the target ~~device~~functional block buffering the data transfers received after issuance of the busy signal until resources become available to service the buffered data transfers.

11. (Currently Amended) The method as set forth in claim 7, wherein determining service guarantees comprises:

mapping the first transaction stream to data channels of components between ~~an~~the initiator ~~device~~functional block and the target ~~device~~functional block;

converting performance guarantees of selected data channels of the mapped first transaction stream such that ~~the~~ guarantees of the data channels are aligned to be uniform in units; and

aggregating the guarantees of the data channels for the first transaction stream.

12. (Currently Amended) The method as set forth in claim 11, wherein aggregating comprises a function selected from the group consisting of summing the guarantees of the data channels of the first transaction stream, selecting the maximum guarantees of the data channels of the first transaction stream, and selecting the minimum guarantees of the data channels of the first transaction stream.

13. (Original) The method as set forth in claim 11, wherein the guarantees are selected from the group consisting of quality of service guarantees, performance guarantees, bandwidth guarantees, latency guarantees, maximum outstanding request guarantees and maximum variance in service latency guarantees.

14. (Currently Amended) A method for communicating data between functional blocks in a computing device, comprising:

establishing at least one firstthread identifier, each firstthread identifier associating a data transfer with a transaction stream that the data transfer between an initiator functional block and a target functional block are part of;

if the target functional block is unable to accept a first data transfer from the initiator functional block, the target functional block issuing a busy signal identified by the firstthread identifier;

storing in a buffer one or more data transfers received by the target functional block after issuance of the busy signal until resources become available to service the buffered data transfers, the amount of buffer space sufficient to buffer any transfers that arrive after the busy signal is asserted, wherein an interface between the initiator functional block and the target functional block does not block data transfers of other threads; and

mapping a data flow from the initiator functional block to the target functional block to a thread first transaction stream indicated by the its unique firstthread identifier to meet a service guarantee on a per firstthread identifier basis within a multiple threading system.

15. (Currently Amended) The method as set forth in claim 14, wherein the target functional block issues a busy signal a determined number of clock cycles after the target functional block determines that it is unable to accept athe first data transfer from anthe initiator functional block.

16. (Original) The method as set forth in claim 14, further comprising the target functional block receiving no more than a determined number of data transfers after issuance of the busy signal.

17. (Currently Amended) The method as set forth in claim 14, further comprising determining service guarantees for at least one transaction stream between a plurality of initiator functional blocks and ~~the~~target functional blocks.

18. (Currently Amended) The method as set forth in claim 17, wherein determining service guarantees comprises:

mapping the first transaction stream to data channels of components between ~~an~~the initiator ~~device~~functional block and the target ~~device~~functional block;

converting performance guarantees of selected data channels of the mapped first transaction stream such that ~~the~~ guarantees of the data channels are aligned to be uniform in units; and

aggregating the guarantees of the data channels for the first transaction stream.

19. (Currently Amended) The method as set forth in claim 18, wherein aggregating comprises a function selected from the group consisting of summing the guarantees of the data channels of the first transaction stream, selecting the

maximum guarantees of the data channels of the first transaction stream, and selecting the minimum guarantees of the data channels of the first transaction stream.

20. (Currently Amended) A communication apparatus, comprising:

at least two functional blocks within a multiple threading system, wherein an initiator functional block communicates with a target functional block by establishing a connection; and

a bus coupled to each of the functional blocks and configured to carry a plurality of signals, wherein the plurality of signals comprises a first~~thread~~ identifier configured to associate a data transfer with a transaction stream between the initiator functional block and the target functional block, and a credit signal identified by the first~~thread~~ identifier, the credit signal issued by the target functional block to indicate how many data transfers the target functional block can accept, wherein the initiator functional block associated withholds issuance of data transfers associated with the first~~thread~~ identifier if the credit signal indicates that the target functional block can accept no data transfers, and the bus being non-blocking, via the use of credit signals, to enable a determination of service guarantees for transaction streams between initiator functional blocks and target functional blocks.

21. (Original) The apparatus as set forth in claim 20, wherein the busy signal comprises a signal that is maintained active when the target functional block is unable to accept data transfers.

22. (Original) The apparatus as set forth in claim 20, wherein the busy signal comprises a credit signal comprising a number of credits that indicate how many data transfers the target functional block can accept.

23. (Original) The apparatus as set forth in claim 22, wherein the number of credits is decremented for each active data transfer and incremented upon freeing up of resources for further data transfers.

24. (Currently Amended) The apparatus as set forth in claim 22, wherein the credit signal is generated by maintaining the credit signal in an active state for a number of clock cycles corresponding to the number of credits.

25. (Previously Presented) The apparatus as set forth in claim 22, wherein the credit signal comprises a multi-bit coded signal indicative of the number of credits.

26. (Currently Amended) The apparatus as set forth in claim 20, wherein the at least one transaction stream is non-blocking enabling determination of

service guarantees for transaction streams between the initiator functional blocks and the target functional blocks.

27. (Original) The apparatus as set forth in claim 20, wherein the target functional block further comprises a buffer to receive data transfers issued by the initiator functional block after issuance of the busy signal by the target functional block and before receipt of the busy signal by the initiator functional block.

28. (Original) The apparatus as set forth in claim 27, wherein service guarantees are determined by mapping the transaction stream to data channels of components between an initiator device and target device, converting performance guarantees of selected data channels of the mapped transaction stream such that the guarantees of the data channels are aligned to be uniform in units, and aggregating the guarantees of the data channels for the transaction stream.

29. (Original) The apparatus as set forth in claim 28, wherein aggregating comprises a function selected from the group consisting of summing the guarantees of the data channels of the transaction stream, selecting the maximum guarantees of the data channels of the transaction stream, and selecting the minimum guarantees of the data channels of the transaction stream.

30. (Original) The apparatus as set forth in claim 26, wherein the guarantees are selected from the group consisting of quality of service guarantees, performance guarantees, bandwidth guarantees, latency guarantees, maximum outstanding request guarantees and maximum variance in service latency guarantees.

31. (Currently Amended) A communication apparatus, comprising:

at least two functional blocks within a multiple threading system, wherein an initiator functional block communicates with a target functional block by establishing a connection;

a bus coupled to each of the functional blocks and configured to carry a plurality of signals, wherein the plurality of signals comprises at least one first~~thread~~ identifier configured to associate a data transfer with a transaction stream that the data transfer between an initiator functional block and a target functional block are part of; wherein if the target functional block is unable to accept a data transfer from the initiator functional block, then the target functional block issuing a busy signal identified by the first~~thread~~ identifier and buffering data transfers received after issuance of the busy signal until resources become available to service the buffered data transfers; and

a buffer coupled to the target functional block, ~~the~~a size of the buffer sufficient to buffer any number of data transfers that arrive ~~on~~in the transaction stream after the busy signal is asserted~~; and~~, wherein the bus implements a mapping algorithm to map a data flow of the transaction stream and aggregate

service guarantees from components between the initiator functional block and the target functional block.

32. (Currently Amended) The apparatus as set forth in claim 31, wherein the target functional block issues a busy signal a determined number of clock cycles after the target functional block determines that it is unable to accept a first data transfer from ~~an~~the initiator functional block.

33. (Original) The apparatus as set forth in claim 31, further comprising the target functional block receiving no more than a determined number of data transfers after issuance of the busy signal.

34. (Currently Amended) The apparatus as set forth in claim 31, further comprising the target functional block determining service guarantees for at least one transaction stream between a plurality of initiator functional blocks and ~~the~~ target functional blocks.

35. (Currently Amended) The apparatus as set forth in claim 34, wherein determining service guarantees comprises:

mapping the transaction stream to data channels of components between ~~an~~the initiator ~~device~~functional block and target ~~device~~functional block;

selectively converting determined service guarantees of data channels of components of the mapped transaction stream such that ~~the~~ guarantees of the data channels are aligned to be uniform in units; and

aggregating the guarantees of the data channels for the transaction stream.

36. (Original) The apparatus as set forth in claim 35, wherein aggregating comprises a function selected from the group consisting of summing the guarantees of the data channels of the transaction stream, selecting the maximum guarantees of the data channels of the transaction stream, and selecting the minimum guarantees of the data channels of the transaction stream.

37. (New) The apparatus as set forth in claim 31, wherein the first identifier is a connection ID.

38. (New) The apparatus as set forth in claim 31, wherein the first identifier is a thread ID.

```
Set     Items   Description
S1    1596789   FLOW??? OR PROGRESS??? OR CONTINU????? OR COURSE? ? OR
            MOVEMENT OR SEQUENCE OR SUCCESSION
S2     158612   S1(3N)(DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR
            KB??)
S3      11129   S2(5N)(THREAD? ? OR CONNECT??? ? OR LINK? ? OR
            ATTACH???? OR PATH? ? OR CIRCUIT? ?)
S4     222020   (INIT?????? OR SOURCE OR START??? OR BEGIN???? OR
            ORIGIN?? OR FIRST)(3N)(NODE? ? OR FUNCTIONAL()BLOCK? ? OR
            COMPONENT? ? OR CONSTITUENT? ? OR ELEMENT? ?)
S5         75   S3(5N)S4
S6     129248   (FINAL OR TARGET? ? OR TERMINAL? ? OR DESTINATION? ? OR
            END??? OR LAST)(3N)(NODE? ? OR FUNCTIONAL()BLOCK? ? OR
            COMPONENT? ? OR CONSTITUENT? ? OR ELEMENT? ?)
S7         13   S5(5N)S6
S8          0   (MAP???? OR TRAC??? OR PLAN???? OR PROJECT??? OR
            STRATEG??? OR PREPLAN????)(5N)S7
S9          7   S7 AND (BUSY OR OCCUP??? OR UNAVAIL????)
S10         0   S9 AND (SERVICE? ? OR QOS OR FUNCTION? ? OR
            TRANSACTION? ? OR JOB? ? OR TASK? ?) AND (GUARANTEE? ? OR
            AGREEMENT? ? OR CONTRACT? ? OR GUARANTY)
S11         0   S9 AND (GUARANTEE? ? OR AGREEMENT? ? OR CONTRACT? ? OR
            GUARANTY)
S12   1207555   MAP???? OR TRAC??? OR PLAN???? OR PROJECT??? OR
            STRATEG??? OR PREPLAN????
S13    992505   DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR KB??
S14   1652909   THREAD? ? OR CONNECT??? ? OR LINK? ? OR ATTACH???? OR
            PATH? ? OR CIRCUIT? ?
S15    188924   BUSY OR OCCUP??? OR UNAVAIL????
S16   1083341   SERVICE? ? OR QOS OR FUNCTION? ? OR TRANSACTION? ? OR
            JOB? ? OR TASK? ?
S17    141124   GUARANTEE? ? OR AGREEMENT? ? OR CONTRACT? ? OR GUARANTY
S18         7   (S1 AND S4 AND S5 AND S12 AND S13 AND S14 AND S15 AND
            S16 AND S17) NOT S9
S19    115046   (S16 AND S17) NOT (S9 OR S18)
S20     21003   S19 AND S14 AND S15
S21         1   S20 AND S4 AND S5
S22     15420   S20 AND S12 AND S13
S23       104   (S22 AND IC=(G06F-009/46 OR G06F-015/163 OR G06F-009/54
            OR G06F-009/00)) NOT (S21 OR PD=(20000309:20030309) OR
            PD=(20030309:20060728))
S24        11   S23 AND (S16 OR S17)/TI
S25         8   S23 AND (ASIC? ? OR DRAM? ? OR FPGA? ? OR VLSI? ? OR
            (SYSTEM? ?(3N)(CHIP? ? OR CIRCUIT? ?))) NOT S24
? show files
File 348:EUROPEAN PATENTS 1978-2006/ 200630
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20060727,UT=20060720
         (c) 2006 WIPO/Univentio
```

25/3,K/3 (Item 3 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00759423
Method for providing multiple grades of service with protection against overloads in a shared resources system
Verfahren zur Versorgung mehrerer Service -Stufen mit Schutz gegen Uberlastung in einem gemeinsamen Betriebsmittelsystem
Methode pour fournir plusieurs niveaux de service avec protection contre les surcharges dans un systeme a ressources partagees
PATENT ASSIGNEE:
AT&T Corp., (589370), 32 Avenue of the Americas, New York, NY 10013-2412, (US), (applicant designated states: DE;ES;FR;GB;IT)
INVENTOR:
Choudhury, Gagan Lal, 602 Randall Way, Aberdeen, New Jersey 07747, (US)
Whitt, Ward, 86 Hill Top Road, Basking Ridge, New Jersey 07920, (US)
Leung, Kin K., 10 Rainford Road, Edison, New Jersey 08820, (US)
LEGAL REPRESENTATIVE:
Johnston, Kenneth Graham et al (32381), AT&T (UK) Ltd. 5 Mornington Road, Woodford Green Essex, IG8 OTU, (GB)
PATENT (CC, No, Kind, Date): EP 714062 A2 960529 (Basic)
EP 714062 A3 971029
APPLICATION (CC, No, Date): EP 95308137 951114;
PRIORITY (CC, No, Date): US 344268 941123
DESIGNATED STATES: DE; ES; FR; GB; IT
INTERNATIONAL PATENT CLASS (V7): **G06F-009/46** ; H04L-012/56; H04Q-011/04;
ABSTRACT WORD COUNT: 214

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB96	1202
SPEC A	(English)	EPAB96	24181
Total word count - document A			25383
Total word count - document B			0
Total word count - documents A + B			25383

Method for providing multiple grades of service with protection against overloads in a shared resources system
Verfahren zur Versorgung mehrerer Service -Stufen mit Schutz gegen Uberlastung in einem gemeinsamen Betriebsmittelsystem
Methode pour fournir plusieurs niveaux de service avec protection contre les surcharges dans un systeme a ressources partagees
INTERNATIONAL PATENT CLASS (V7): **G06F-009/46** ...

...ABSTRACT form steady-state distribution. The techniques allow each customer to obtain an appropriate grade of **service** and protection against overloads from other customers. Each customer is a source of a series of requests, and is assigned "upper-limit" (UL) and " **guaranteed** -minimum" (GM) "bounds" on its requests. The upper limit bound puts an upper limit on the number of requests from that customer that can be in **service** at any time. The **guaranteed** -minimum bound **guarantees** that there will always be available resource units in the resources to serve a specified...

...form steady-state distribution. The BPC computes the normalization constants by first constructing the generating **function** (or z-transform) of the normalizing constant and then numerically inverting the generating **function** . (see image in original document)

...SPECIFICATION A2

Field of the Invention

The invention relates to efficiently providing multiple grades of **service** , including protection against overloads, for multiple customers sharing limited resources, and, more particularly, to (i...

...new customer is admitted or allowed to use the resources with a desired grade of **service** , (ii) adjusting the capacities of the resources in the face of changing customer demand, and...

...sharing problems generally involve one or more "resources", each containing multiple resource "units" which provide **service** to multiple "customers". Each customer is a source of a series or stream of "requests...

...in many physical settings, for example, in communication networks and in computer systems. In a **circuit** -switched telecommunications network, the resources may be " **links** " and the resource units may be " **circuits** " on these **links** , while the customers may be associated with different " **services** " (e.g., voice, **data** , video and facsimile), and the requests may be "calls ". Calls often require **circuits** simultaneously on several different **links** , with the specific **links** depending on the origin and destination of the call. For some **services** , calls require multiple **circuits** on each **link** .

A **circuit** -switched telecommunication network example is depicted in Fig. 1. In this figure, there are 5 nodes, 101-105, and 5 **links** , 111-115, with **link** "i" having K(sub(i)) **circuits** . (In this case, the nodes play no role.) This network serves multiple customers, with the customer being characterized partly by the set of **links** or "route" that the customer requires. For example, in the setting of Fig. 1 there might be six routes requiring the following subsets of **links** : (Table omitted) There might be many more customers, because there can be different customers on the same route. For example, some customers may require only one **circuit** on each **link** , whereas other customers may require multiple **circuits** on each **link** . To be more concrete, certain kinds of **data** **service** calls require 6 **circuits** on each **link** , while standard voice calls require only one **circuit** on each **link** . In this setting a customer might represent a class of calls, either voice or **data** , on a particular route. Thus, there would be a total of 12 customers, with one **data** customer and one voice customer on each of the six routes. The six customers on routes 1, 3 and 6 thus all use **link** 111 and share the K(sub 1) available **circuits** there. When a **data** call is attempted on route 6, it requires 6 **circuits** on each of the **links** 111, 113 and 115. If, at the time of the attempt, sufficient free **circuits** are not available on any of these **links** , then the call is blocked. In contrast, a voice call on route 1 requires only one **circuit** on **link** 111. Unless special measures are taken, as provided by this invention, the **data** calls on route 6 will clearly experience much higher blocking than the voice calls on...

...in this setting the network provider wants to be able to provide appropriate grades of **service** to the different customers, including protection against overloads from other customers.

In a broadband integrated- **services** digital network (B-ISDN) supported by the asynchronous transfer mode (ATM) technology, the resources may...

...while the customers may be prospective "users" of the network who wish to establish a **connection** and the customer requests may be ATM cells, bursts of ATM cells or required "effective bandwidths" associated with

bursts of ATM cells within an established **connection** . Fig. 1 could also apply to this B-ISDN example. If the critical resources are...

...then the nodes 101- 105 are the resources. More generally, both nodes 101-105 and **links** 111-115 might be important resources. Since B-ISDN networks are intended for a broad range of **services** , it is important to allow customers to have very different traffic characteristics and very different the quality of **service** received in terms of a customer's "request blocking probability ", i.e., the long-run...

...based control does not depend on the number of requests from each customer currently in **service** . Here we consider situations in which this extra **information** is available to the resource provider. It is therefore possible to look for controls which exploit this additional **information** when it is available.

Moreover, many schemes, such as the rate-based multiple-class access...

...state distributions; e.g., see J. S. Kaufman, "Blocking in a Shared Resource Environment", IEEE **Transactions** on Communications, volume COM-29, pages 1474-1481, 1981, but good algorithms for general coordinate...of others, but it is also important to be able to provide "different grades of **service** ." Some customers may be willing to pay more to have lower blocking probabilities and stronger...

...of resource units than another tends to have a higher blocking probability. Different grades of **service** can be achieved by complete partitioning, but what is needed are "more efficient" ways to provide different grades of **service** that allow sharing.

When considering potential schemes for providing multiple grades of **service** with protection against overloads, other associated problems arise. First, the resource provider wants to "assess the costs of providing given grades of **service** ", so that an effective "pricing scheme" can be developed. There are different ways to represent...

...by each customer. However, when there are customers with very different requirements and grades of **service** , careful analysis may be required to determine the average capacity used by each customer. Crude...

...the true costs. With any sharing policy that is used to provide different grades of **service** , the **service** provider would thus like to be able to calculate the average capacity used by each customer.

A successful scheme for efficiently providing multiple grades of **service** should address the problem of "real-time customer admission control". For given limited resources, the...

...determine whether or not each prospective new customer can be admitted. With multiple grades of **service** , the resource provider needs to determine whether the new customer can be given his desired grade of **service** , with all previously admitted customers still receiving their previously determined grades of **service** .

Any technique for providing different grades of **service** presumably imposes bounds on the requests submitted by admitted customers. An efficient scheme for "enforcing...

...the level of customer demand often changes. There may be growing or declining demand for **service** . There may also be a temporary reallocation of demand in the face of "resource failure...

...options to respond to resource failure. As indicated above, customers might be able to receive **service** from an alternative set of resources. In some cases, this will require additional capacity that...

...provided quickly enough. Then the resource provider needs a way to provide the best possible **service** in the face of an uncontrollable resource failure. If customers who were using a failed...

...How to manage and control access to a resource and efficiently provide multiple grades of **service** with protection against overloads?

How to provide **service** with different grades of **service** to a given set of customers using resources with given capacities?

How to assess the "cost" (in terms of resource usage, capacity, etc.) of providing a given grade of **service** ?

How to determine in real time whether or not each prospective customer with a desired grade of **service** can be (or should be) admitted?

How to determine the new capacities of the resources...

...to a resource failure?

Summary of the Invention

The present invention provides different grades of **service** to customers sharing a resource, and also provides protection against overloads. Each customer is assigned "upper-limit" (UL) and " **guaranteed** -minimum" (GM) "bounds" on its requests. The upper limit bound puts an upper limit on the number of requests from that customer that can be in **service** at any time. The **guaranteed** -minimum bound **guarantees** that there will always be available resource units in the resources to serve a specified...

...of Shared Finite Storge in a Computer Network Node Environment Under General Traffic Conditions", IEEE **Transactions** on Communications, volume COM-28, pages 992-1003, 1980, leads to a coordinate-convex sharing...

...much broader context.

While UL bounds can easily be enforced if the resource provider keeps **track** of the number of requests from each customer in **service** , the GM bounds are more difficult to enforce. This new invention includes an "efficient" scheme...

...finite-source input, i.e., requests submitted from a fixed finite population.

Customer grades of **service** can also be specified in another way, in particular, via "conditional blocking requirements", which are...

...requirements are met.

In order to efficiently manage shared resources with the proposed grades of **service** , the present invention both approximately and exactly solves the resource-sharing model with these grades of **service** in effect, using a "blocking probability computer" process.

As indicated above, the resource-sharing model...

...the desired blocking probabilities can be directly expressed in terms of normalization constants (or partition **function** values) appearing in the product-form steady-state distribution. The process computes the normalization constants by first constructing the generating **function** (or z-transform) of the normalizing constant and then numerically inverting the generating **function** .

Since numerical inversion of the generating **function** can be difficult, the present invention includes a process to facilitate the inversion. The main difficulty is that the required computation grows

exponentially in the dimension of the generating **function** . The dimension of the generating **function** is in turn equal to the number of resources plus the number of separate linear...

...consideration. These very lightly loaded resources are eliminated from the model before constructing the generating **function** .

After performing the preliminary analysis with the normal approximation, the generating **function** for the model with all remaining resources is formed. An explicit expression for this generating **function** has been determined as a **function** of the customer parameters.

Next the effective dimension of the generating **function** is reduced. A "conditional decomposition scheme" is used to determine a good order (sequence) for inverting the variables of the multidimensional generating **function** . The conditional decomposition scheme often can significantly reduce the effective dimension of the generating **function** . Indeed, this conditional decomposition step always works with the UL and GM bounds to reduce...speedup.

With the arrangement just described, the blocking probability computer determines whether proposed grades of **service** can be provided to a given set of customers using resources with given capacities.

In...

...each new prospective customer can be admitted to a resource with a desired grade of **service** . The first step in the process is to determine the desired grade of **service** for the prospective customer, yielding proposed customer traffic parameters and UL and GM bounds. The...

...be met, then the resource provider admits the new customer with the desired grade of **service** . If all blocking requirements cannot be met, then the blocking probability computer is used to determine if a lower grade of **service** is feasible.

The present invention also uses the blocking probability computer to make appropriate capacity...

...customer demand. Given a new set of prospective customers with specified resources and grades of **service** , the blocking probability computer finds appropriate resource capacities. A normal approximation scheme is first used...

...diverted customers.

Brief Description of the Drawing

Fig 1. is a representation of a conventional **circuit** -switched telecommunication network with which the present invention can operate.

Fig. 2 is a block...

...requirements.

Fig. 5 is a flow diagram illustrating the process for generating alternative grades of **service** when a requested grade of **service** is not feasible.

Fig. 6 is a flow diagram illustrating the process by which traffic...

...Detailed Description

There are three principal aspects of our technique for providing multiple grades of **service** with protection against overloads in shared resources, namely: (A) real-time customer admission control (B...

...bounds, and (C) response to resource failures. All three exploit the upper-limit (UL) and **guaranteed** -minimum (GM) bounds and the blocking probability computer. Each of the three aspects are discussed...

...From time to time, customers from customer pool 202 wish to be admitted to (receive **service** from) resource 201. The decision whether or not to admit the customer is made by...

...controller 203, which signals a switch 206 to close or open, so as to either **connect** a customer arrival from customer pool 202 to resource 201, or to deny a customer arrival access to resource 201. Admission controller 203 also determines the grade of **service** , including the upper limit (UL) and **guaranteed** minimum (GM) bounds. When a customer wishes to be admitted to resource 201, the customer...controller 203. At this time, the customer indicates his desired traffic parameters and grade of **service** , including blocking requirements. Admission controller 203 then invokes blocking probability computer 204 to determine whether...

...is constructed, which includes all existing customers as well as the new customer. The relevant **data** about existing customers is obtained by admission controller 203 from a customer database 205. If...

...p)) are positive integers.)

Next, in step 302, the requirements of all customers currently in **service** are determined. The specific requirements are described in conjunction with the explanation of Step 306 below. To determine the requirements of all customers currently in **service** , the resource provider would update a customer database upon each new customer arrival or departure. The customer database contains a record of all customers in **service** including the parameters of their grade of **service** .

Given that grades of **service** are provided by using UL and GM bounds, it is necessary to enforce these traffic bounds on an ongoing basis for each customer in **service** . This **function** is performed in step 303, and, is described in more detail below. This step is...

...the process to move to step 305. If the new event is the completion of **service** of an existing customer, then that customer **data** must be removed from the customer database, and the process returns to step 302. Although not specifically shown in Fig. 3, if a customer wishes to renegotiate their grade of **service** , this is treated as a completion of **service** request. Note that the customer or the system may monitor the customer's actual request...

...a higher or lower request arrival rate. Customers who want to change their grade of **service** can be thought of as a **service** completion followed immediately by a new arrival.

If the result of step 305 indicates a new arrival, then, in step 306, the desired grade of **service** and the resulting requirements and traffic bounds are determined. The grade of **service** includes a characterization of the request traffic. The standard characterization of a customer's request...

...described below.)

Alternatively, the traffic for customer j can be characterized by arrival-rate and **service** -rate **functions** (lambda)(sub(j))(k) and (mu)(sub(j))(k). Then (lambda)(sub(j))(k) is...

...when k requests from customer j are active and (mu)(sub(j))(k) is the **service** (completion) rate of requests when k requests from customer j are active. The standard case...

...mu)(sub(j)), corresponding to a constant arrival rate for the customer

and a constant **service** rate per active request. An arrival rate of the form (lambda)(sub(j))(k) =(alpha...

...from a finite source. The blocking probability computer applies with general state-dependent arrival and **service** rates as well as in the standard case.

The standard case corresponds to having peakedness...

...each i and j. In order to have the blocking probability computer perform efficiently with **guaranteed** minimum bounds, it is assumed that b(sub(ij)) is either b(sub(j)) or...

...e., the positive entries of b(sub(j)) assume a constant value.)

The grade of **service** also includes blocking requirements. First, there is the nominal blocking requirement, which is the desired negotiated rates. Customers may also specify their grade of **service** directly through the UL and GM bounds. If so, these bounds may still need to...

...the blocking probability computer is used to determine if one or more alternative grades of **service** can be provided. The modified requirements might, for example, have lower arrival-rate **function** , lower upper-limit bound or lower **guaranteed** -minimum bound.

After the alternative grades of **service** are proposed in step 310, a determination is made in step 311 as to whether...

...returns to step 306, where the new parameters are used to determine the grade of **service** . On the other had, if the result in step 311 is "No", the new customer...

...the process returns to step 304 to await a new customer event.

The grade of **service** determination performed in step 306 of Fig. 3 is described in further detail in Fig. 4. The process begins in step 401, when the available **service** options are communicated to the customer. The system may provide a fixed finite set of pre-specified grades of **service** at pre-specified prices, or it may design a new grade of **service** tailor-made for each customer. Customers are informed of the policy.

Next, in step 402 the system receives input from the customer. Given a specification of the proposed grade of **service** , the resource provider computes an estimate of the cost and tells the customer the price...

...is acceptable to the customer.

The requirement modification procedure used to propose alternative grades of **service** in step 310 of Fig. 3 is described in more detail in Fig. 5. This...

...with the same arrival rate that is feasible. The procedure to generate alternative grades of **service** stops in step 505.

The two alternatives generated in steps 502 and 503 are offered...

...of Fig. 3).

As indicated in step 303 of Fig. 3, providing multiple grades of **service** with UL and GM bounds requires that these bounds be enforced on an ongoing basis...

...efficiently enforcing the traffic bounds is shown in Fig. 6.

Advantageously, this process checks the **guaranteed** -minimum bounds in an efficient manner. In particular, the computational complexity at each request event...resources

r - number of customers
K(sub(i)) - capacity of resource i
L(sub(j)) - **guaranteed** -minimum bound on the requests from customers j
U(sub(j)) - upper-limit bound on...

...of F(sub(i)) is the total number of units minus the number required by **guaranteed** minimum bounds, i.e., (see image in original document)
The process of Fig. 6 waits...

...step 603. If it is determined in step 603 that a customer-j request completes **service** and departs, variables n(sub(j)) and F(sub(i)) are updated as follows:
(i...

...complete-sharing and trunk reservation.
We now discuss ways to determine the cost of providing **service** to a new customer. There are several possible interpretations for cost of providing **service** to a new customer, even if we focus only on the capacity used on each resource. Clearly, the minimum capacity used is the **guaranteed** -minimum bound GM, and the maximum capacity used is the upper limit bound UL. It...

...current customers plus the new customer can be met. The marginal expected cost of providing **service** to this customer on this resource is the difference between these two capacity levels. The...Paper 4.4b.3, and L. E. N. Delbrouck, "A Unified Approximate Evaluation of Congestion **Functions** for Smooth and Peaky Traffic", IEEE **Transactions** on Communications, volume 29, pages 85-91, 1981. "Peakedness" is defined as the ratio of...

...associated infinite-capacity resource. For the case of Poisson arrivals, the number of requests in **service** has a Poisson distribution when there is no capacity limit. Since the variance equals the...

...more general setting.
The basic idea is to use a linear state-dependent arrival-rate **function** (lambda)(sub(j))(k)=(alpha)(sub(j))+k(beta)(sub(j)), since this arrival process, known as a BPP process, produces a number of **busy** servers in an infinite-capacity resource distributed as Pascal for (beta)(sub(j))>0 and...

...determined by matching the mean and variance of the number of class-j requests in **service** at an arbitrary time in an infinite-capacity resource, i.e., a system without any...

...as indicated by A. E. Eckberg (cited above). These parameters can also be estimated from **data** .
Having obtained (alpha)(sub(j)) and (beta)(sub(j)), we wish to compute the blocking...

...step is to compute m(sub(j)), the mean number of class - j requests in **service** in the actual system with capacity constraints. It can be shown that m(sub(j...

...sub(j)) - GM bound on requests from customer j
N(sub(j)) - number of units **guaranteed** for customer j, N(sub(j)) = L(sub(j))b(sub(j))
K = (K(sub...

...lightly loaded resources is described in further detail below.
Next, in step 703, the generating **function** of the normalization constant associated with the product-form steady-state distribution is formed. ...n(sub(r))) and n(sub(j)) is the number of customer-j requests in **service** . The steady-state distribution is then (Formula omitted) where g(K,U,N) is the...

...in original document) (see image in original document) (lambda)(sub(j))(k) the arrival-rate **function** and (mu)(sub(j))(k) the **service** rate **function** . Then the normalization constant is (see image in original document) The generating fraction of the...

...dot)(center dot)(center dot) ,x(sub(r))) are vectors of complex variables. The generating **function** in Equation 4 has the form (see image in original document) where (see image in...

...mu)(sub(j)) and (rho)(sub(j)) = (lambda)(sub(j))/(mu)(sub(j)), the generating **function** F(sub(j))(x) is given by: (Formula omitted)
In the so-called binomial and...

...beta)(sub(j)), and still (mu)(sub(j))(k) = k(mu)(sub(j)), the generating **function** F(sub(j))(x) in Equation 16 is given by: (Formula omitted)
For the special...

...sub(j)), f(sub(j))(n(sub(j))) = (rho)n(sub(j))j the generating **function** F(sub(j))(x), in Equation 16 is given by: (Formula omitted)
The closed-form...

...sub(j)) (x) in Equations 17 through 19 make it easier to calculate the generating **function** values in Equation 15, and thus in Equation 14. However, even in the general case...

...complex to solve directly. This difficulty is primarily due to the dimension of the generating **function** formed in step 703 being too large. When the dimension is too large, a good conditional decomposition is sought to reduce the effective dimension of the generating **function** . The procedure for finding a good conditional decomposition to achieve dimension reduction is described below...

...described in more detail below.
If the result in step 705 is "YES", the generating **function** is inverted in step 707. The inversion can be done in different ways. One effective...

...of Applied probability, volume 4, pages 719-740, 1994.
Suppose that a p-dimensional generating **function** (see image in original document) is to be inverted. It is assumed that conditional decomposition...
...to p one-dimensional inversions recursively.
To represent the recursive inversion, let the partial generating **functions** be (see image in original document) where z(sub(j)) = (z(sub 1),z(sub...

...inversion with j = 2 is needed. This process goes on until at step p the **function** on the righthand side becomes the p-dimensional generating **function** and is explicitly computable.
Shown below is the inversion formula at the j(sup(th...j, because then more computation is done to achieve the same accuracy.
When the inverse **function** is a probability, the aliasing error e(

sub(j)) in Equation 25 can easily be...

...therefore the aliasing error e(sub(j)) may also be arbitrarily large. Thus, the generating **function** is scaled in each step by defining a scaled generating **function** as (Formula omitted) where (alpha)(sub(0j)) and (alpha)(sub(j)) are positive real numbers. This scaled generating **function** is inverted after choosing (alpha)(sub(0j)) and (alpha)(sub(j)), so that the errors...

...noted that the aliasing error in the numerical inversion is controlled by scaling the generating **function** at each step. In the j(sup(th) step of a p-dimensional inversion, a scaled generating **function** is defined in Equation 26. This scaled generating **function** is inverted after choosing (alpha)(sub(0j)) and (alpha)(sub(j)) so that the errors ...

...j))(K(sub(j))), or its fastest growing term, exploiting the structure of the generating **function** .

For the case of the complete sharing (CS) policy with Poisson arrivals, the scaled generating **function** is (see image in original document) and the scaled normalization constant is (see image in **service** -rate **functions** will be described. Only the one-dimensional case will be described in detail. The multidimensional...

...of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions", IEEE **Transactions** on Communications , volume COM-28, pages 992-1003, 1980. This case can be reduced to...

...sharing (CS) policy has been assumed. In the UL case, the form of the generating **function** is the same as a CS generating **function** with more resources. In this case, the CS scaling can be extended in a straightforward...

...UL policy with the upper limits equal to the capacities minus the sum of the **guaranteed** minima of all other classes. For the combined UL and GM case, as an approximation...

...are the minimum of the given upper limit and the heuristic one based on the **guaranteed** minima of all other classes.

There is another heuristic that applies to generating **functions** of any form, and thus is applicable to the general state-dependent arrival and **service** rates. Let, (see image in original document) with (see image in original document) Then (alpha...

...original document)

Turning now to several techniques that can advantageously be used as options in **connection** with the present invention, it is to be noted first that if two or more...

...number of customer classes is (see image in original document)

If the p-dimensional generating **function** of interest can be written as (see image in original document) then it can be...

...the finite sum.

The inversion formula in each dimension is a weighted sum of generating **function** values evaluated over equidistant points along the circumference of a circle. The weights are complex...

...have constant amplitude. As the capacities K(sub(i)) grow, the amplitude of the generating **function** typically becomes unevenly distributed along the circumference of the circle. There are several

local maximum...have constant amplitude, it is only necessary to consider the relative amplitude of the generating **function** values.) If all the relative maximum points can be identified, and then only those points...

...is developed for a single resource and then it is extended. Consider the scaled generating **function** in the case of complete sharing with Poisson arrivals, i.e., (Formula omitted) In the...

...1)K(sub 1). Let G*((theta)) be G(z(sub 1)) expressed as a **function** of (theta), i.e., (see image in original document) Note that the amplitude of G...

...be exploited with multiple resources, but the situation is more complicated. Now the scaled generating **function** is given by (see image in original document) and the scaled normalization constant is (see ...

...sub(i)) for i < p, it is necessary to cope with the partially inverted generating **function** g(sup((j-1))(z(sub(j-1)),K(sub(j))) for which the **functional** form is not known. Hence, the maximum points are not known, so that it is...

...the location of the maximum points is the same as if the partially inverted generating **function** has the same **functional** form as in Equation 55 usually works and gives good computational savings.
In order to...Btj in Equation 65 for the modified model in which the class-j arrival-rate **function** is changed from (lambda)(sub(j))(m) to (lambda)(sub(j))(m) (identical with) (lambda...

...is a model of the same general form. For the model with linear arrival-rate **function** , this approach to computing call blocking was pointed out by Z. Dziong and J. W. Roberts, in "Congestion Probabilities in a **Circuit** -Switched Integrated **Services** Network", Performance Evaluations, volume 7, pages 267-284, 1987, at page 273.
Different expressions are...

...fixed, and omit the i and j subscripts.
It is assumed that the arrival-rate **function** is linear, i.e., (lambda)(k) = (alpha)+k(beta). As indicated above, this includes the... were no capacity constraints, then the mean and variance of the number of requests in **service** would be (Formula omitted) Since each request uses b resource units, the associated mean and...

...For this purpose, let (phi)(x) be the standard (mean 0, variance 1) normal density **function** and let (PHI)(x) be its cumulative distribution **function** .
Assuming that the occupancy for a customer can be approximated by the conditional normal variable...

...through the sum in the inversion formula
However, it virtually never happens that the generating **function** can be factored into separate components with no common variables, because this corresponds to having...

...conditional decomposition". First, select d variables that will be inverted and see if the remaining **function** of p - d variables can be expressed as a product of factors with no two...

...exploited for UL and GM policies to drastically reduce the effective

dimension. Consider the generating **function** displayed in Equation 14. The generating **function** there is directly expressed in terms of factors. Since the factor G(sub(j))(z...

...sub(j)), rewrite Equation 82 as (see image in original document) The overall remaining generating **function** is (see image in original document) where G(sub(j)) (z,U(sub(j)),N...each customer at each resource by the blocking experienced by that customer elsewhere. This approximation **strategy** , together with a process to solve a single resource, leads to a nonlinear system of...

...S. P. Chung and K. Ross "Reduced Load Approximations for Multi-Rate Loss Networks", IEEE **Transactions** on Communications volume 41, pages 1222-1231, 1993; F. P. Kelly, "Loss Networks", Annals of Applied Probability, volume 1, pages 319-378, 1991; and W. Whitt, "Blocking When **Service** is Required from Several Facilities Simultaneously", AT&T Technical Journal, volume 64, pages 1807-1856...

...with the concept of peakedness and normal approximations; see W. Whitt "Heavy-Traffic Approximations for **Service** Systems with Blocking", AT&T Bell Laboratories Technical Journal, volume 63, pages 689-708, 1984...
...for customer j at submodel k, assumed to be based directly on the original model **data** . Equation 87 exploits the independence approximation, because it arises by assuming that the probability of... Since in general there are state-dependent arrival rates, new reduced state-dependent arrival-rate **functions** (lambda)*kj for customer j at submodel k are formed by letting (see image in original document)
The reduced arrival-rate **function** (lambda)*kj depends on the submodel blocking probabilities B(sub(1j)), and the submodel blocking probabilities in turn depend on the arrival-rate **functions** used in the submodels. Hence, it is necessary to find a fixed-point solution to...

...original document) Ways to compute means and variances for individual classes were described above in **connection** with the normal-approximation algorithm for eliminating lightly loaded resources. Finally, resources that are most...

...monitor 902. Customers can indicate their desire to either increase or decrease their grade of **service** . Traffic load monitor 902 can indicate that the current capacity needs to be either increased...These are bounds on the number of requests, not the number of resource units. To **guarantee** a minimum grade of **service** in case of extreme overload of all other customers, customer j may have a lower...
...adjustment is needed, step 1003 determines the available resources and customer requirements. Based on these **data** , the search procedure attempts to find a good resource capacity by the following four major...

...by M(sub(j)) and V(sub(j)) respectively, of the number of resource units **occupied** by each customer j. Let the average and the variance of the number of resource units **occupied** by all customers be M and V, which are in turn obtained by M= (SIGMA...m(sub(j)) and the variance v(sub(j)) of the number of requests in **service** from each customer j.
(c) Define a set C to be (1,2,...r). Set...m(sub(j)) and the variance v(sub(j)) of the number of requests in **service** from each customer j.
(c) Define a set C to be (1,2,...,r). Set...1101 detects whenever one or more of the resources fails. In a telecommunications system with **links** as resources, the **link** status is typically monitored in the switches within the network. A **link** failure can be detected therein by, for example, a loss of signal.

When a resource...

...as described above in Section B).
Periodically, traffic load measurement system 1102 sends traffic load **data** to a traffic load database 1104 within COC 1110. Thus, at the time of resource...

...being met by the failed resource. For this purpose, traffic diversion controller 1105 obtains necessary **data** about availability of alternative resources from a routing database 1106. Then, traffic diversion controller 1105...

...which is described below. Traffic diversion controller 1105 also determines appropriate upper limit (UL) and **guaranteed** minimum (GM) bounds to protect both the original traffic and the diverted traffic on these...

...can be met by the remaining resources. In a resource-sharing system, each customer request **occupies** certain units of different resources for a period of time. When a resource fails, customer...

...common. In the example of a circuited-switch telecommunication network in Fig. 1, when a **link** (i.e., resource) fails for some reason such as a fiber cut, calls can be routed through other **links** , bypassing the failed **link** . In many cases, the total number of resource units needed to satisfy a specific customer...
...is higher than in normal conditions; e.g., because the alternative route often has more **links** . Nevertheless, the use of alternative resources provides great flexibility in efficiently sharing the resources among...

...future customer requests that originally demand the failed resource, without adversely affecting the quality of **service** provided to other customers? (It is assumed that those customer requests receiving **service** from the failed resource at the time of the failure are lost, but some of...for the resource. For example, events such as earthquakes, flooding, and hurricanes damaging a telephone **link** can also cause a large number of calls to be made to the disaster area, thus putting more demand for the failed **link** . Such a potentially large increase of customer requests for the failed resource, which are now diverted to the remaining resources, can seriously degrade the quality of **service** for the original customers. Moreover, there is no **guaranteed** minimum grade of **service** for the diverted customers.
The central idea of this aspect of the invention is to...

...proportion of the load, the procedure adjusts the UL and GM parameters to try to **guarantee** the same grade of **service** for the diverted traffic of each customer. Finally, two new sets of UL bounds are...
...diverted traffic, can be accommodated as much as possible, while protecting the satisfactory grade of **service** for customer 3 and 4 requests, the original traffic. One can view the diversion of...sub(d)) can be found by normal approximations as follows. Let the number of circles **occupied** by the original traffic be approximated by a normally distributed random variable, N(m(sub...

...refined by using the blocking probability computer.
Until the square resource is put back in **service** , the system diverts each new customer 1 and 2 request to the circle resource for **service** , while the UL and GM traffic bounds for each customer, as well as the new ...

...requests are served in case of failure of the needed square resource, the grades of **service** for all customers are also **guaranteed** and protected by the UL and GM bounds.
The basic principle of protecting both the...

...to a specific system, a telecommunication network is considered. This network can be a current **circuit** -switched network or a future B-ISDN network based on the ATM technology. A high-level flowchart for traffic diversion in response to **link** failures in the network is outlined in Figure 12 and the diversion procedure is discussed...

...Consider a telecommunication network such as shown in Fig. 1, with a number of communication **links** , which are the resources in question. To ensure a high degree of reliability, there often are multiple routes (i.e., a **path** consisting of multiple **links**) from one switch (node) to another in the network. Thus, each call can possibly be...and adaptive in their response to failures as follows. In non-adaptive routing, when one **link** of a route from one switch to another fails, the entire route must be replaced...

...between the two end switches (referred to as the failure-end switches) of the failed **link** . As a result, calls can be routed according to the original route until reaching a...

...end switch. Then, the calls are directed to the new alternative route, bypassing the failed **link** to the other failure-end switch. Finally, the call routing proceeds from there along the...

...original route, until reaching their destination. It is possible that there are multiple new routes **connecting** the failure-end switches. In this case, each call in question can take one of...
...that low priority routes are attempted for routing the call when high priority routes are **busy**), or other state-dependent dynamic scheme can be used for this purpose.
If the network...

...to other alternative routes and to enforce the UL and GM traffic bounds for performance **guarantee** and protection. In this case, the traffic diversion should be carried out on the basis...

...may be more efficient for the failure-end switches to divert traffic from the failed **link** to alternative routes **connecting** the switches and to enforce the associated UL and GM bounds for the diverted calls...

...telecommunication network has a centralized operations center (COC) for traffic monitoring, diversion and other maintenance **functions** , block 1110 in Fig. 11. Also assume that critical **information** for traffic diversion is available at the COC. This **information** includes: a) the expected call load for each origin-destination switch pair in the network, b) the preferred routes for calls for each switch pair, and c) the grade of **service** in terms of blocking probabilities, the contracted (offered) load, and the UL and GM bounds for each call class on each **link** . With the contracted load in the network, these UL and GM parameters are specified and enforced to **guarantee** a minimum grade of **service** and to protect against overload of each class of calls.
Assuming that a **link** i carries r classes of calls, let the contracted load for the **link** be denoted by (rho)(sup AND)(sub(o)) = (rho)(sup AND)1o,(rho)(sup AND...

...ro). In addition, let the UL and GM bounds in terms of the number of

circuits for the contracted traffic for link i be U(sub(o))(identical with) (U1o, U2o,..., Uro) and L(sub(o))(identical...

...on (rho)(sup AND)(sub(o)), U(sub(o)) and L(sub(o)) for each link i, the COC pre-computes the binding parameters c(sub(j)) and d(sub(j...

...the arrival peakedness of class j calls and b(sub(ij)) is the number of circuits occupied by a class j call on link i.

The traffic diversion process starts with step 1201 of the flowchart of Fig. 12...

...amount of traffic load of different call classes (i.e., the carried load) on each link emerging from the switch and the fraction of calls blocked (i.e., the blocking probabilities). Each switch reports the carried load and blocking probabilities for each of its links to the COC periodically and the COC saves the data for future referencing.

When a switch detects a link failure in step 1202 by loss of signal from the link , it reports the failure to the COC. Let switch A and B be the failure-end switches for the failed link . After receiving the notification of the failure, the COC retrieves in step 1203, the most current data of the carried load and blocking probabilities for the failed link . It is assumed that the COC can develop a good estimate of the offered load...

...carried load. The estimated load could simply be the original contracted load (for which no data are needed) or the most recent estimate of carried load, but it could be a more complicated combinations of these and other historical data . For simplicity, here it is assumed that only the most recent carried load estimate is...

...there be r classes of calls sent from switch A to B via the failed link and let the call classes be indexed by 1,2,...,r. Furthermore, let the carried load and the blocking probability for each call class j for the failed link be denoted by (phi)(sub(j)) and P(sub(j)), respectively, for j=1,2,...,r. Following the data retrieval, the COC estimates the amount of offered load of class i calls on the failed link from switch A to B, (rho)jf, by dividing (phi)(sub(j)) by 1-P...

...that this offered load is expected to be different from the contracted load for the link because of traffic fluctuation. The COC may thus elect to adjust (rho)jf, e.g., replace (rho)jf by the minimum or some other function of the observed (carried) load and the contracted load for that class. Let (rho)(sub...

...alternative route exists, then all alternative routes from switch A to B, bypassing the failed link , have been considered and loaded with a portion of the diverted traffic. In this case...

...12 stops in step 1207, although it cannot fully divert all traffic from the failed link without degrading the grade of service for the original traffic in the network.

If a new alternative route exists, so that...

...rho)1d,(rho)2d,...,(rho)rd), that the alternative route can accept from the failed link . In particular, this method uses a proportional approach to divert traffic of all classes; that is, the same proportion of calls from all classes is diverted from the failed link to an alternative route. Furthermore, as an approximation step in the analysis, it is assumed here that the original traffic for all classes on all links of the contemplated route are mutually independent Thus, in the model, only the diverted calls require the simultaneous

possession of all **links** of the alternative route. (Another approach is to consider the whole network with all call...

...is determined by the following steps:

(a) Let the chosen alternative route consist of J **links**, indexed by 1,2,...,J. The COC estimates the offered load for each call class on each of these **links** by dividing the carried load by one minus the blocking probability for the call class on the **link**. (The carried load and blocking proability on each **link** are updated and recorded at the COC periodically.) As with diverted traffic, this may be the minimum or some other **function** of the contracted and measured offered load for the **link**. For simplicity, each **link** is assumed to have the same number r of call classes. Let the offered load for one of the J **links** be denoted by (rho)(sub(o)) = ((rho)1o,(rho)2o,...,(rho)ro).

(b) For the...

...J+1)r classes of calls (customers), where each resource represents one of the J **links** in the alternative route, and the j(sup(th) and (J+1)(sup(st) set...

...classes of requests correspond to the original and diverted traffic on the j(sup(th) **link** of the route, respectively. As stated above, it is assumed that the original traffic classes on all **links** are mutually independent and only the diverted calls require the simultaneous possession of all J **links**. Let (alpha)(sub(U)) and (alpha)(sub(L)) denote the respective upper and lower bounds for the proportion of traffic to be diverted from the failed **link**. Initially, set (alpha)(sub(L))=0 and (alpha)(sub(U))=1.

(c) Set (alpha)=((alpha...

...o,i)) and U(sub(d,i)) for the original and diverted traffic on each **link** i have been found. The procedure stops. Otherwise, proceed with step e) below.

(e) For each **link** i of the alternative route, set (gamma)(sub(i)) to be the ratio of expected **circuit** occupancy of the diverted traffic to that of the contracted traffic on **link** i. That is, (Formula omitted) Let the UL and GM bounds for the diverted traffic on **link** i be U(sub(d))(identical with)(U1d, U2d,..., Urd) and L(sub(d))(identical ...

...the pre-determined per-call-class binding parameters for the UL and GM bounds for **link** i described above. For each call class j, set Ujd to be the minimum of...

...Formula omitted) and K(sub(i)), where K(sub(i)) is the total number of **circuits** in **link** i. In addition, set Ljd to be the minimum of the positive integer closest to...

...f) For each call class j, choose the minimum among all Ujd's for all **links** i of the alternative route. Without introducing additional notation, let Ujd denote such minimum. Similarly, for each class j, obtain the minimum Ljd among the Ljd's for all **links** i. (The UL and GM traffic bounds Ujd and Ljd can be enforced for class j calls on every **link** of the alternative route.)

(g) In order to provide further protection against overload, obtain two...

...UL bounds (denoted by U(sub(o,i)) and U(sub(d,i)) for every **link** i) for the original traffic and the diverted traffic by normal approximations as follows. For each **link** i of the alternative route, let the number of **circuits** **occupied** by the original (contracted)

traffic on **link** i be approximated by a normally distributed random variable, N(m(sub(o)),(sigma)2o...

...traffic bounds. Then, the COC computes the remaining traffic to be diverted from the failed **link** in step 1209, and the process returns to step 1204 to divert the remaining traffic, if any. This procedure continues until all traffic is successfully diverted from the failed **link** or all alternative routes have been examined and loaded with certain amount of the diverted...

...enforced by the switches on the alternative route. More precisely, the bounds associated with a **link** can be observed by the switch that sends traffic onto the **link** . In contrast, the per-call-class UL and GM bounds and the new UL bound...

...know the corresponding minimum of the UL and GM bounds for various call classes or **links** for proper traffic enforcement for the diverted calls.

One way to enable efficient traffic enforcement is to keep the **link** failure known to only the failure-end switches and the COC. As a result, other switches in the network can continue to route traffic as if the **link** failure did not occur. When a call needs to be routed from switch A to...

...can identify if the call is a diverted call (i.e., one requires the failed **link**). If so, switch A makes sure that the UL and GM traffic bounds are satisfied...routing algorithm of the network requires the use of the loading status of the failed **link** , and if it is not available for the diverted traffic directly, then such **information** can be assembled by the failure-end switches, by keeping **track** of the loading status of diverted traffic among the established alternative routes.

With this traffic diversion method, one can view that a failed **link** is partially or even fully "replaced" by the established alternative routes, because the GM parameters **guarantee** a minimum grade of **service** for the diverted traffic on the alternative routes, and the new UL parameters protect the...

...from possible overload of each other.

In fact, this traffic diversion procedure can handle multiple **link** failures and switch failures. Note that a switch failure can be regarded as a case of multiple (simultaneous) **link** failures for all **links** emerging from the switch. For the case of multiple **link** failures that do not occur simultaneously, the diversion procedure can work well, because traffic can be diverted from one failed **link** at a time on a first-come-first-served basis. If a particular **link** is involved in several alternative routes for different failed **links** , the **link** can be loaded with traffic diverted from multiple failed **links** . As a result, the **link** carries an additional set of call classes diverted from each failed **link** . In this situation, the diverted traffic from various failed **links** will have their respective UL and GM traffic bounds, which can be enforced separately by...

...the traffic load to its neighboring switches (i.e., traffic load on each of the **links** emerging from the switch) as suggested above, but also the amount of traffic routed to...

...network without a COC. In this case, each switch monitors and keeps the traffic load **data** locally. When a **link** failure occurs, a distributed algorithm, such as the distributed asynchronous Bellman-Ford algorithm (see Bertsekas and Gallager, **Data** Networks, Prentice-Hall, 1992,

pp.404-410), can be used to identify the shortest alternative...

...alternative route is found, the failure-end switch A is responsible for collecting the traffic **data** needed for the diversion from the switches on the alternative route. With the traffic **data** , switch A performs the traffic diversion in the same way as the COC does. As...

...If switch A finds that there is traffic remaining to be diverted from the failed **link** , it initiates further traffic diversion on the next alternative route. The traffic enforcement by the...
...original traffic can continue to be done by the switches that send traffic onto the **links** . In addition, the traffic bounds for the diverted traffic can be enforced by the failure...

...CLAIMS blocking probability requirements of said existing customers;
determining the blocking probability requirements and grade of **service** desired by each of said new customers;
determining if said new customers' requirements can be...

...shared resource by said new customers and said existing customers by
numerically inverting the generating **function** of said normalization constant of said resource sharing model.
2. A method of controlling admission...

...a shared resource, in which said new customers can be provided with multiple grades of **service** , including protection against overloads from existing customers and other new customers, said method comprising the steps of
retrieving stored **information** defining the requirements of said existing customers, and obtaining the requirements and grade of **service** desired by each of said new customers;
responsive to said retrieved **information** , determining if said new customer's requirements can be satisfied without violating the requirements of...
...said new customer to be admitted to said shared resource with said desired grade of **service** only if said new customers requirements can be satisfied
wherein said last mentioned determining step...

...shared resource by said new customers and said existing customers by
numerically inverting the generating **function** of said normalization constant of said resource sharing model.
4. A method of controlling admission...

...which said existing customers and said new customer can be provided with multiple grades of **service** , each of said grades of **service** being defined by a traffic requirement and a blocking requirement, said method comprising the steps of
obtaining **information** indicating (a) the grade of **service** of said existing customers and (b) the grade of **service** desired by said new customer;
responsive to said grade of **service** desired by said new customer, assigning UL and GM bounds on the usage of said shared resource to said new customer,
determining if said new customer's grade of **service** can be provided without violating the requirements of said existing customers, and
if said new customer's grade of **service** can be provided, allowing said new customer to be admitted to said shared resource with said desired grade of **service** ,
wherein said determining step includes computing the blocking

probability for said new customer, and determining...

...a prospective new customer can be admitted to a resource with a desired grade of **service** , comprising the steps of
determining the desired grade of **service** for the prospective customer,
determining if the desired grade of **service** can be met considering both the prospective customer and all existing customers,
if the desired grade of **service** can be provided, admitting the new customer with the desired grade of **service** , and
if the desired grade of **service** cannot be met, then determining if a lower grade of **service** is feasible.

6. A method of adjusting the capacity of a shared resource so as...

...probabilities for usage of said shared resource by said customers by numerically inverting the generating **function** of said normalization constant of said resource sharing model.

7. A method of diverting customer...

...by said diverted customers and said existing customers are calculated by numerically inverting the generating **function** of said normalization constant of said resource sharing model.

8. A technique for providing different grades of **service** and protection against overloads to customers sharing a resource, comprising the steps of
assigning each customer upper-limit (UL) and **guaranteed** -minimum (GM) bounds on its requests, said upper limit bound putting an upper limit on the number of requests from that customer that can be in **service** at any time, and said **guaranteed** -minimum bound guaranteeing that there will always be available resource units in the resources to...

...which operates by
(a) directly expressing said model in terms of normalization constants (or partition **function** values) appearing in the product-form steady-state distribution,
(b) computing the normalization constants by constructing the generating **function** of the normalizing constant and
(c) numerically inverting the generating **function** .

9. The method of claim 8 wherein said numerically inverting step includes
using a Fourier...

...said last mentioned step, eliminating lightly loaded resources from said model before constructing said generating **function** .

11. The invention defined in claim 8 or 9 further including reducing the effective dimension of the generating **function** by conditional decomposition and inverting the variables of the multidimensional generating **function** in an appropriate order.

12. The method of claim 11 further including determining, after said...

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 714 062 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.05.1996 Bulletin 1996/22

(51) Int Cl.[6]: G06F 9/46

(21) Application number: 95308137.9

(22) Date of filing: 14.11.1995

(84) Designated Contracting States:
DE ES FR GB IT

(30) Priority: 23.11.1994 US 344268

(71) Applicant: AT&T Corp.
New York, NY 10013-2412 (US)

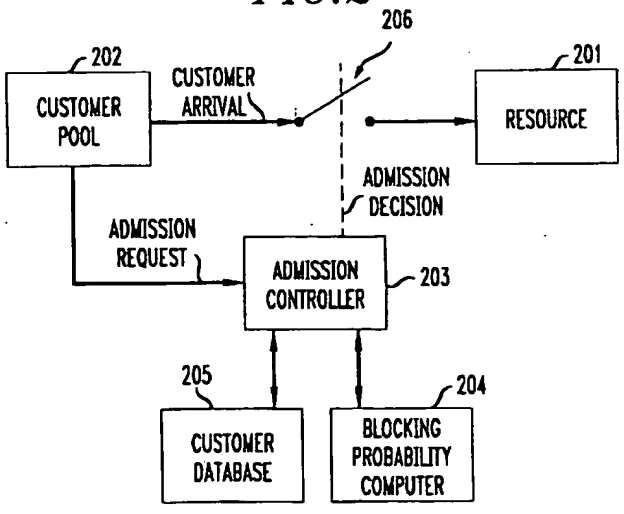
(72) Inventors:
- Choudhury, Gagan Lal
Aberdeen, New Jersey 07747 (US)
- Whitt, Ward
Basking Ridge, New Jersey 07920 (US)
- Leung, Kin K.
Edison, New Jersey 08820 (US)

(74) Representative: Johnston, Kenneth Graham et al
AT&T (UK) Ltd.
5 Mornington Road
Woodford Green Essex, IG8 OTU (GB)

(54) Method for providing multiple grades of service with protection against overloads in a shared resources system

(57) Techniques for (a) controlling admission of customers to a shared resource, (b) adjusting the capacity of a resource in light of new customer demand, and (c) diverting usage from a failed resource to alternative resources, each use a "blocking probability computer" (BPC) to solve a resource-sharing model that has a product-form steady-state distribution. The techniques allow each customer to obtain an appropriate grade of service and protection against overloads from other customers. Each customer is a source of a series of requests, and is assigned "upper-limit" (UL) and "guaranteed-minimum" (GM) "bounds" on its requests. The upper limit bound puts an upper limit on the number of requests from that customer that can be in service at any time. The guaranteed-minimum bound guarantees that there will always be available resource units in the resources to serve a specified number of requests from that customer. The desired blocking probabilities are directly expressed in terms of normalization constants appearing in the product-form steady-state distribution. The BPC computes the normalization constants by first constructing the generating function (or z-transform) of the normalizing constant and then numerically inverting the generating function.

FIG. 2



EP 0 714 062 A2

Description

Field of the Invention

The invention relates to efficiently providing multiple grades of service, including protection against overloads, for multiple customers sharing limited resources, and, more particularly, to (i) controlling when a prospective new customer is admitted or allowed to use the resources with a desired grade of service, (ii) adjusting the capacities of the resources in the face of changing customer demand, and (iii) responding to resource failure.

Background of the Invention

Resource-sharing problems generally involve one or more "resources", each containing multiple resource "units" which provide service to multiple "customers". Each customer is a source of a series or stream of "requests". Each customer request requires a number of units from each resource, which may be zero, one, or greater than one, and may be different for different customers, but which is the same for different requests of the same customer. If all requirements can be met upon arrival of a new request, then the new request is admitted, and all required resource units are reserved or held throughout the request holding time. Otherwise, the request is not admitted, and is said to be "blocked".

Resource sharing as described above takes place in many physical settings, for example, in communication networks and in computer systems. In a circuit-switched telecommunications network, the resources may be "links" and the resource units may be "circuits" on these links, while the customers may be associated with different "services" (e.g., voice, data, video and facsimile), and the requests may be "calls ". Calls often require circuits simultaneously on several different links, with the specific links depending on the origin and destination of the call. For some services, calls require multiple circuits on each link.

A circuit-switched telecommunication network example is depicted in Fig. 1. In this figure, there are 5 nodes, 101-105, and 5 links, 111-115, with link "i" having K_i circuits. (In this case, the nodes play no role.) This network serves multiple customers, with the customer being characterized partly by the set of links or "route" that the customer requires. For example, in the setting of Fig. 1 there might be six routes requiring the following subsets of links:

Route	Links
1	111
2	112
3	111, 112
4	113, 115
5	114, 115
6	111, 113, 115

There might be many more customers, because there can be different customers on the same route. For example, some customers may require only one circuit on each link, whereas other customers may require multiple circuits on each link. To be more concrete, certain kinds of data service calls require 6 circuits on each link, while standard voice calls require only one circuit on each link. In this setting a customer might represent a class of calls, either voice or data, on a particular route. Thus, there would be a total of 12 customers, with one data customer and one voice customer on each of the six routes. The six customers on routes 1, 3 and 6 thus all use link 111 and share the K_1 available circuits there. When a data call is attempted on route 6, it requires 6 circuits on each of the links 111, 113 and 115. If, at the time of the attempt, sufficient free circuits are not available on any of these links, then the call is blocked. In contrast, a voice call on route 1 requires only one circuit on link 111. Unless special measures are taken, as provided by this invention, the data calls on route 6 will clearly experience much higher blocking than the voice calls on route 1. Moreover, in this setting the network provider wants to be able to provide appropriate grades of service to the different customers, including protection against overloads from other customers.

In a broadband integrated-services digital network (B-ISDN) supported by the asynchronous transfer mode (ATM) technology, the resources may be "switches " and other network facilities, the resource capacity (units) may be the "bandwidth" available at these network facilities, while the customers may be prospective "users" of the network who wish to establish a connection and the customer requests may be ATM cells, bursts of ATM cells or required "effective bandwidths" associated with bursts of ATM cells within an established connection. Fig. 1 could also apply to this B-ISDN example. If the critical resources are the switches, then the nodes 101-105 are the resources. More generally, both nodes 101-105 and links 111-115 might be important resources. Since B-ISDN networks are intended for a broad

range of services, it is important to allow customers to have very different traffic characteristics and very different requirements. In particular, requests from different customers may use different numbers of resource units. In this context, admission control is a well known but challenging problem of current interest (see J. W. Roberts, "Performance Evaluation and Design of Multiservice Networks," COST 224 Final Report, Commission of the European Communities, Luxembourg, 1992).

The resource provider is faced with two fundamental problems: The first is the obvious fact that resource units are expensive to provide, so that it is important to have no more capacity than necessary. The second is the probabilistic nature of the problem. The submission of customer requests and the holding times of these requests are uncertain events that fluctuate over time. Therefore, the actual requirements of the customers cannot be known in advance. However, the pattern of customer requests and request holding times can be predicted in a probabilistic sense. Indeed, it is known that probabilistic or stochastic models may be used to characterize the customer requirements. In this uncertain environment with limited resources, some blocking of customer requests becomes inevitable unless the resource capacity greatly exceeds demand. It is thus customary to characterize the quality of service received in terms of a customer's "request blocking probability ", i.e., the long-run proportion of requests that are blocked in a particular operating regime. Each customer wishes to have its request blocking probability be suitably low. Request blocking probabilities that are too high fail to satisfy customer requirements. On the other hand, request blocking probabilities that are too low mean that more capacity has been provided than is needed.

In this probabilistic setting, the resource provider needs to: (i) determine the admissibility of a given set of customers for given resource capacities, (ii) determine appropriate resource capacities for a given set of customers, and (iii) determine how to respond to temporary unavailability of one or more resources because of resource failure. These problems can be considered when each customer's request traffic is characterized (probabilistically) and each customer's blocking probability requirements are given. When all customers have full access to the resources, i.e., when a "complete-sharing policy" is used, there are known methods to calculate the blocking probabilities for each class in a model of the system under consideration. The resource provider can then check to see if the prevailing customer-request blocking probabilities meet the requirements. The resource-provider can use these calculations to determine whether or not to admit new customers, to determine appropriate capacities, and to determine how to respond to resource failures. However, existing methods for computing these blocking probabilities encounter severe difficulties as the model gets large. Thus, there has remained a need to be able to solve much larger models of the standard form than could be solved up to now.

In addition to having difficulty when the model becomes large, existing methods have difficulty computing blocking probabilities when the request arrival processes have exceptionally high or low variability. The standard representation of an arrival process in the resource-sharing model is a "Poisson" process. However, in may applications the arrival process of customer requests is significantly more or less variable than a Poisson process, and these alternative forms of arrival process variability significantly affect the customer request blocking probabilities. For example, highly variable arrival processes routinely arise in overflow traffic, as occurs when there is alternative routing in a telecommunication network. Thus, there is a need for an effective way to characterize non-Poisson arrival process variability and accurately determine the customer request blocking probabilities.

Moreover, the resource provider actually is strongly motivated "not" to use the complete-sharing policy, for which known methods often apply. An important feature of many emerging resources is the presence of different kinds of customers. In this setting, it is very important to protect the customers from each other. If all customers are allowed full access to the resource, then one or more customers may actually submit requests at very high rates, much above their negotiated rates, which can cause other customers in other classes to experience unacceptably high blocking probabilities.

Thus, some way is needed to "protect customers from overloads by other customers. " One way to do this is to "partition " the resources into separate portions dedicated to each customer, but this tends to be inefficient, because the benefits of sharing are lost. The overall capacity required in each resource tends to become unacceptably large with complete partitioning.

Another possible control scheme that allows some sharing is the rate-based multiple-class access-control scheme, such as that described in U. S. Patent No. 5,274,644 issued on December 28, 1993, to Berger, Milito and Whitt. That scheme regulates the requests admitted from different customers, but it is based solely on the pattern of requests. It does not take into account the holding times of these requests. More generally, it does not use the current state of the resources. In particular, that rate-based control does not depend on the number of requests from each customer currently in service. Here we consider situations in which this extra information is available to the resource provider. It is therefore possible to look for controls which exploit this additional information when it is available.

Moreover, many schemes, such as the rate-based multiple-class access control scheme of the above-cited Berger et al. patent only regulate the flow of customer requests. The resource provider also needs to know whether or not to allow customers to send requests. The customer admission problem is also of major concern.

Instead of complete sharing or complete partitioning, it is natural to consider various "constraints" on the allowable

states in the resources that still allow some sharing. One such scheme is "trunk reservation". For example, with two classes, trunk reservation blocks one of the two classes whenever the total free capacity falls below a specified threshold. This scheme protects one of the two classes against overloads from the other, but it does not protect both classes. More generally, with n classes, a natural generalization of trunk reservation has the first k classes blocked if the total free capacity falls below a specified k^{th} threshold, where there are n-1 threshold values. However, just as in the two-class case, all classes are not protected with this scheme.

Moreover, using known techniques, it is relatively difficult to calculate the customer blocking probabilities with the "trunk- reservation" scheme. The trunk-reservation scheme causes the standard model to lose the nice product-form structure which makes it possible to analyze the complete-sharing and complete-partitioning policies. Other constraints also tend to cause severe problems in computing the blocking probabilities. It is known that a large class of constraints, leading to the so-called "coordinate-convex" sharing policies, do possess product-form steady-state distributions; e. g., see J. S. Kaufman, "Blocking in a Shared Resource Environment", IEEE Transactions on Communications, volume COM-29, pages 1474-1481, 1981, but good algorithms for general coordinate convex policies have not been developed. Algorithms for computing customer request blocking probabilities with constraints have previously been developed in only a few very special cases.

With different customers, it is not only important to protect against overloads of others, but it is also important to be able to provide "different grades of service." Some customers may be willing to pay more to have lower blocking probabilities and stronger protection against overloads from others, while other customers would prefer to have higher blocking probabilities and weaker protection against overloads from others at a lower price.

Even achieving similar blocking probabilities is difficult, because different customers may require different numbers of resource units for each request. A customer whose requests require a larger number of resource units than another tends to have a higher blocking probability. Different grades of service can be achieved by complete partitioning, but what is needed are "more efficient" ways to provide different grades of service that allow sharing.

When considering potential schemes for providing multiple grades of service with protection against overloads, other associated problems arise. First, the resource provider wants to "assess the costs of providing given grades of service", so that an effective "pricing scheme" can be developed. There are different ways to represent such costs. One important way is to determine the average capacity on each resource used by each customer. However, when there are customers with very different requirements and grades of service, careful analysis may be required to determine the average capacity used by each customer. Crude estimates based only on average usage may fail to accurately represent the true costs. With any sharing policy that is used to provide different grades of service, the service provider would thus like to be able to calculate the average capacity used by each customer.

A successful scheme for efficiently providing multiple grades of service should address the problem of "real-time customer admission control". For given limited resources, the resource provider needs to be able to determine whether or not each prospective new customer can be admitted. With multiple grades of service, the resource provider needs to determine whether the new customer can be given his desired grade of service, with all previously admitted customers still receiving their previously determined grades of service.

Any technique for providing different grades of service presumably imposes bounds on the requests submitted by admitted customers. An efficient scheme for "enforcing" these bounds is needed. Since checking has to be done for each customer request, computational efficiency is of serious concern.

Over time, the level of customer demand often changes. There may be growing or declining demand for service. There may also be a temporary reallocation of demand in the face of "resource failure." In some resource failure situations, customers can be served by assigning them to alternative resources, but this increases the customer demand on these alternative resources. In response, the resource provider may have the opportunity to add capacity to these other resources. In face of such changing customer demand, the resource provider needs a way to determine the amount of capacity needed to meet this demand. This is the "capacity adjustment problem".

The resource provider often will have a more complicated set of options to respond to resource failure. As indicated above, customers might be able to receive service from an alternative set of resources. In some cases, this will require additional capacity that can be immediately provided at other resources, possibly at extra expense. However, in other cases, additional capacity cannot be provided quickly enough. Then the resource provider needs a way to provide the best possible service in the face of an uncontrollable resource failure. If customers who were using a failed resource can be assigned to alternative resources, then ways are needed to protect the original customers on these other resources from the diverted demand. At the same time, the resource provider would like to protect the diverted customers as well.

There are many methods to identify alternative resources available to serve customers that were using a failed resource. In some settings, the alternative resources may be evident. For example, the system may contain only two resources, with each serving as a backup for the other. In other settings, there may be an automatic procedure in place to dynamically reallocate demand to new resources, as is the case with schemes for alternative routing of blocked calls in telecommunications networks. Another possibility is that a special procedure may need to be invoked by a central

controller in the event of resource failure, as in the algorithms for finding alternative routing arrangements developed by Mansour and Nguyen (U.S. patent No. 5,058,105). Where there is no centralized control, a distributed algorithm may be needed to first inform all resources that a failure has taken place, and then to set up appropriate alternative routes (resource assignments).

Regardless of the method used to generate alternative resource assignments, there is a need to provide some protection for both the original and diverted customers on the remaining resources.

In summary, the resource provider is faced with many problems:

How to manage and control access to a resource and efficiently provide multiple grades of service with protection against overloads?

How to provide service with different grades of service to a given set of customers using resources with given capacities?

How to assess the "cost" (in terms of resource usage, capacity, etc.) of providing a given grade of service?

How to determine in real time whether or not each prospective customer with a desired grade of service can be (or should be) admitted?

How to determine the new capacities of the resources needed to satisfy a new set of customer demands?

How to respond to a resource failure?

Summary of the Invention

The present invention provides different grades of service to customers sharing a resource, and also provides protection against overloads. Each customer is assigned "upper-limit" (UL) and "guaranteed-minimum" (GM) "bounds" on its requests. The upper limit bound puts an upper limit on the number of requests from that customer that can be in service at any time. The guaranteed-minimum bound guarantees that there will always be available resource units in the resources to serve a specified number of requests from that customer. The process provides what can be called significant separation with sharing and outperforms complete partitioning and complete sharing.

Variations on the UL and GM bounds are also allowed. Alternative bounds can be based on other linear constraints on the number of active requests of different customers. These alternative bounds include UL bounds for subsets (classes) of customers.

The use of UL and GM bounds, which has been described in a special and limited context by Kamoun and Kleinrock in "Analysis of Shared Finite Storge in a Computer Network Node Environment Under General Traffic Conditions", IEEE Transactions on Communications, volume COM-28, pages 992-1003, 1980, leads to a coordinate-convex sharing policy, so that the resulting resource-sharing model has a product-form steady-state distribution. In accordance with the present invention, we have discovered an effective method for determining blocking probabilities in this model, so that the various problems faced by the resource-provider can be solved, and the UL/GM bounds applied in a much broader context.

While UL bounds can easily be enforced if the resource provider keeps track of the number of requests from each customer in service, the GM bounds are more difficult to enforce. This new invention includes an "efficient" scheme for enforcing both the GM and UL bounds.

The UL and GM bounds may either (a) be provided directly by the customer or (b) be determined by the resource provider to satisfy other requirements. It is assumed that the customer provides its blocking requirements and a characterization of its desired traffic. The request blocking requirement is the maximum allowed request blocking probability, assuming that all customers submit requests according to their negotiated traffic parameters.

A customer's traffic is characterized by the request arrival rate, the average request holding time, and the number of units needed on each resource. In addition, a "burstiness parameter", which describes the variability of the customer's request stream, is also allowed. The standard assumption is a Poisson arrival process. The burstiness parameter accounts for arrival processes substantially more or less bursty than a Poisson process. In addition, the customer requests can have state-dependent arrival rates or batch arrivals. A state-dependent arrival rate includes the important case of a finite-source input, i.e., requests submitted from a fixed finite population.

Customer grades of service can also be specified in another way, in particular, via "conditional blocking requirements", which are blocking requirements conditional on the other customers being in some pattern of overload. For one example, there can be a conditional blocking requirement given that any one other customer is in arbitrary overload. For another example, there can be a conditional blocking requirement given that all other customers are in overload,

i.e., given that each other customer is submitting traffic at a rate above its negotiated rate. The UL and GM bounds are used to ensure that the conditional blocking requirements are met.

In order to efficiently manage shared resources with the proposed grades of service, the present invention both approximately and exactly solves the resource-sharing model with these grades of service in effect, using a "blocking probability computer" process.

As indicated above, the resource-sharing model with UL and GM bounds has a product-form steady-state distribution. To surmount the difficulty provided by the extra constraints, the present invention exploits the fact that the desired blocking probabilities can be directly expressed in terms of normalization constants (or partition function values) appearing in the product-form steady-state distribution. The process computes the normalization constants by first constructing the generating function (or z-transform) of the normalizing constant and then numerically inverting the generating function.

Since numerical inversion of the generating function can be difficult, the present invention includes a process to facilitate the inversion. The main difficulty is that the required computation grows exponentially in the dimension of the generating function. The dimension of the generating function is in turn equal to the number of resources plus the number of separate linear constraints. Since the UL and GM bounds typically correspond to a large number of linear constraints, it is necessary to simplify the problem even for a single resource.

In accordance with our process, the first step is to use a "normal approximation scheme" to approximately determine the load on each resource. This preliminary approximate analysis determines if there are some resources that are so lightly loaded that they provide essentially no constraint, so that they can be eliminated from consideration. These very lightly loaded resources are eliminated from the model before constructing the generating function.

After performing the preliminary analysis with the normal approximation, the generating function for the model with all remaining resources is formed. An explicit expression for this generating function has been determined as a function of the customer parameters.

Next the effective dimension of the generating function is reduced. A "conditional decomposition scheme" is used to determine a good order (sequence) for inverting the variables of the multidimensional generating function. The conditional decomposition scheme often can significantly reduce the effective dimension of the generating function. Indeed, this conditional decomposition step always works with the UL and GM bounds to reduce the dimension to at most two more than the dimension with the complete-sharing policy. If there is only a single resource, then the resulting model is almost always solvable.

If there are a substantial number of resources, then the model may still not be solvable after the normal approximation and conditional decomposition steps. If the model is not yet solvable, then a "reduced-load fixed-point approximation process" is used to approximately solve the model. While the general idea of reduced-load fixed-point approximation schemes is known, our invention uses a new subroutine based on the blocking probability computer process to exactly solve the single-resource submodels having UL and GM bounds. The new subroutine may also be used to exactly solve models involving subsets of more than a single resource.

After the three problem-reduction steps are completed, the numerical inversion is performed to calculate the blocking probabilities. This is either done directly, if the model is directly solvable, or indirectly as part of a subroutine within the reduced-load fixed-point approximation scheme. In accordance with our invention, the inversion exploits (a) a Fourier-series method, (b) an effective scaling algorithm especially tailored to the resource-sharing model, (c) truncation, (d) an efficient treatment of multiplicities, and (e) shared-computation of normalization constants when there are many classes and large resource capacities, to obtain significant computational speedup.

With the arrangement just described, the blocking probability computer determines whether proposed grades of service can be provided to a given set of customers using resources with given capacities.

In one specific embodiment of the present invention relating to real-time admission control, the blocking probability computer is used to determine in real time whether each new prospective customer can be admitted to a resource with a desired grade of service. The first step in the process is to determine the desired grade of service for the prospective customer, yielding proposed customer traffic parameters and UL and GM bounds. The blocking probability computer then determines if all blocking requirements (conditional and unconditional) can be met considering both the prospective customer and all existing customers. If all blocking requirements can be met, then the resource provider admits the new customer with the desired grade of service. If all blocking requirements cannot be met, then the blocking probability computer is used to determine if a lower grade of service is feasible.

The present invention also uses the blocking probability computer to make appropriate capacity adjustments to meet changes in customer demand. Given a new set of prospective customers with specified resources and grades of service, the blocking probability computer finds appropriate resource capacities. A normal approximation scheme is first used to find an upper bound on the feasible capacities. Then the capacities are steadily decreased, with the UL and GM bounds changed as necessary. The blocking probability computer is used to evaluate each candidate. The process yields feasible capacities that are as small as can be found by this method.

The present invention also uses the blocking probability computer to determine an effective response to resource

failure, so that customers who were using one of the failed resources can have their demand satisfied on the remaining resources. The blocking probability computer then determines if the new arrangement is feasible. If it is not feasible, then the resource provider considers capacity expansions. If capacity expansions are not possible, then an attempt is made to divert proportions of the traffic on successive sets of alternative resources. The blocking probability computer is used to determine the feasible proportions. Then the UL and GM bounds are used to provide appropriate protection for the original and diverted customers.

Brief Description of the Drawing

Fig 1. is a representation of a conventional circuit-switched telecommunication network with which the present invention can operate.

Fig. 2 is a block diagram of an admission control system arranged in accordance with the principles of the present invention.

Fig. 3 is a flow diagram illustrating the admission control process performed by the admission control system of Fig. 2.

Fig. 4 is a flow diagram illustrating a process for determining new customer requirements.

Fig. 5 is a flow diagram illustrating the process for generating alternative grades of service when a requested grade of service is not feasible.

Fig. 6 is a flow diagram illustrating the process by which traffic bounds are enforced.

Fig. 7 is a flow diagram illustrating the process by which blocking probability computer 204 of Fig. 2 operates.

Fig. 8 is a flow diagram illustrating the process in accordance with the present invention for identifying lightly loaded resources using a normal-approximation technique.

Fig. 9 is a block diagram of a system arranged in accordance with the present invention for making capacity adjustments in a resource-sharing system.

Fig. 10 is a flow diagram illustrating the capacity adjustment process performed in the system of Fig. 9

Fig. 11 is a block diagram of a system arranged in accordance with the present invention for performing traffic diversion in response to a resource failure in a resource-sharing system, and

Fig. 12 is a flow diagram illustrating the traffic diversion process performed in the system of Fig. 11.

Detailed Description

There are three principal aspects of our technique for providing multiple grades of service with protection against overloads in shared resources, namely: (A) real-time customer admission control (B) adjustment of resource capacities and control bounds, and (C) response to resource failures. All three exploit the upper-limit (UL) and guaranteed-minimum (GM) bounds and the blocking probability computer. Each of the three aspects are discussed in turn below. The blocking probability computer is discussed in conjunction with the admission control in Section A.

A. Real-Time Customer Admission Control

A block diagram of an *admission control system* arranged in accordance with the present invention is shown in Fig. 2. A resource 201 (which could have multiple components, each with multiple units), handles arrivals from a source of customers called a customer pool 202. From time to time, customers from customer pool 202 wish to be admitted to (receive service from) resource 201. The decision whether or not to admit the customer is made by an admission controller 203, which signals a switch 206 to close or open, so as to either connect a customer arrival from customer pool 202 to resource 201, or to deny a customer arrival access to resource 201. Admission controller 203 also determines the grade of service, including the upper limit (UL) and guaranteed minimum (GM) bounds. When a customer wishes to be admitted to resource 201, the customer makes an admission request to admission controller 203. At this time, the customer indicates his desired traffic parameters and grade of service, including blocking requirements. Admission controller 203 then invokes blocking probability computer 204 to determine whether the new customer should be admitted, with what UL and GM bounds, and at what price. For this purpose, a mathematical model of the resource-sharing system is constructed, which includes all existing customers as well as the new customer. The relevant data about existing customers is obtained by admission controller 203 from a customer database 205. If the results of the calculations performed by blocking probability computer 204 indicate that all blocking probabilities are less than the blocking requirements, then the new customer can be admitted. This decision is implemented and communicated to the customer by admission controller 203.

A flow diagram depicting an overview of the *admission control process* in accordance with the present invention is given in Fig. 3. The process is initiated by specifying the resources and their capacities in step 301. There are p resources. (Even the case $p = 1$ is of major interest) Resource i has capacity K_i, $1 \leq i \leq p$; i.e., resource i has K_i resource

units. (It is assumed that p, $K_1,...,K_p$ are positive integers.)

Next, in step 302, the requirements of all customers currently in service are determined. The specific requirements are described in conjunction with the explanation of Step 306 below. To determine the requirements of all customers currently in service, the resource provider would update a customer database upon each new customer arrival or departure. The customer database contains a record of all customers in service including the parameters of their grade of service.

Given that grades of service are provided by using UL and GM bounds, it is necessary to enforce these traffic bounds on an ongoing basis for each customer in service. This function is performed in step 303, and, is described in more detail below. This step is not part of the main process flow because we are here concerned with the decision whether or not to admit customers. In contrast, enforcing traffic bounds is concerned with the decision whether or not to admit requests submitted by customers already admitted and permitted to submit requests.

When a new customer event occurs, a "proceed" signal is generated in step 304, causing the process to move to step 305. If the new event is the completion of service of an existing customer, then that customer data must be removed from the customer database, and the process returns to step 302. Although not specifically shown in Fig. 3, if a customer wishes to renegotiate their grade of service, this is treated as a completion of service request. Note that the customer or the system may monitor the customer's actual request traffic and determine that the customer requires different traffic parameters, e.g., a higher or lower request arrival rate. Customers who want to change their grade of service can be thought of as a service completion followed immediately by a new arrival.

If the result of step 305 indicates a new arrival, then, in step 306, the desired grade of service and the resulting requirements and traffic bounds are determined. The grade of service includes a characterization of the request traffic. The standard characterization of a customer's request traffic is via the request arrival rate and the average request holding time. Indeed, it is only necessary to know the product of these parameters, which is called the offered load.

However, it is also possible to have more elaborate traffic characterizations. In addition to the characterization above, there can be a traffic variability parameter, called the "peakedness". The peakedness describes the variability of the request stream. Peakedness is important for treating overflow processes, as occur with alternative routing. (The way to work with peakedness is described below.)

Alternatively, the traffic for customer j can be characterized by *arrival-rate and service-rate functions* $\lambda_j(k)$ and $\mu_j(k)$. Then $\lambda_j(k)$ is the arrival rate of requests when k requests from customer j are active and $\mu_j(k)$ is the service (completion) rate of requests when k requests from customer j are active. The standard case is $\lambda_j(k) = \lambda_j$ and $\mu_j(k) = k\mu_j$, corresponding to a constant arrival rate for the customer and a constant service rate per active request. An arrival rate of the form $\lambda_j(k) = \alpha_j - k\beta_j$ covers the case of input from a finite source. The blocking probability computer applies with general state-dependent arrival and service rates as well as in the standard case.

The standard case corresponds to having peakedness 1. Other peakedness parameters are treated approximately by constructing appropriate state-dependent arrival rates. An effective way to convert peakedness input into state-dependent arrival rates, and then calculate blocking probabilities, is described below.

In step 306, the customer also specifies a *requirements vector* $\bar{b}_j = (b_{1j},...,b_{pj})$. The parameter b_{ij} indicates that each request from customer j requires b_{ij} units of resource i. (It is assumed that b_{ij} is a nonnegative integer for each i and j. In order to have the blocking probability computer perform efficiently with guaranteed minimum bounds, it is assumed that b_{ij} is either b_j or 0; i.e., the positive entries of $\bar{b}_j$ assume a constant value.)

The grade of service also includes *blocking requirements.* First, there is the *nominal blocking requirement,* which is the desired blocking probability for requests from customer j, assuming that all customers submit requests at their nominal rates. Second, there may also be *conditional blocking requirements,* which are blocking probability specifications assuming that one or more other customers are sending requests in excess of their negotiated rates. Customers may also specify their grade of service directly through the UL and GM bounds. If so, these bounds may still need to be altered to meet the blocking requirements. If a customer specifies a UL or GM bound, then the network provider is free to choose any values for these bounds greater than or equal to the bounds specified by the customer.

After step 306 is complete, the *blocking probability computer* is used in step 307 to calculate the blocking probabilities of all customers, old and new, to determine if all blocking requirements can be met. The blocking probability computer is described in more detail below. It also may be necessary for the resource provider to introduce new UL and GM bounds for the new customer in order to meet the blocking requirements, nominal and conditional, of this new customer and the existing customers. The new bound assignment can be done in a manner similar to the way UL and GM bounds are adjusted in the capacity adjustment process, which is described in detail in Section B below.

Next, in step 308 a test is performed to determine if the blocking requirements can be met. If a "YES" result occurs, the new customer is accepted in step 309, and the process returns to step 302. On the other hand, if any computed blocking probability is greater than its requirement, the requirements for admission are not met, and the result of step 308 is a "NO". Then, as indicated in step 310, the blocking probability computer is used to determine if one or more alternative grades of service can be provided. The modified requirements might, for example, have lower arrival-rate function, lower upper-limit bound or lower guaranteed-minimum bound.

After the alternative grades of service are proposed in step 310, a determination is made in step 311 as to whether one of those alternatives is acceptable to the customer. if yes, the process returns to step 306, where the new parameters are used to determine the grade of service. On the other had, if the result in step 311 is "No", the new customer is rejected in step 312, and the process returns to step 304 to await a new customer event.

The grade of service determination performed in step 306 of Fig. 3 is described in further detail in Fig. 4. The process begins in step 401, when the available service options are communicated to the customer. The system may provide a fixed finite set of pre-specified grades of service at pre-specified prices, or it may design a new grade of service tailor-made for each customer. Customers are informed of the policy.

Next, in step 402 the system receives input from the customer. Given a specification of the proposed grade of service, the resource provider computes an estimate of the cost and tells the customer the price in step 405. Ways to estimate the cost are described below. At this point, in step 406, the resource provider determines if the price is acceptable to the customer.

The requirement modification procedure used to propose alternative grades of service in step 310 of Fig. 3 is described in more detail in Fig. 5. This process is initiated in step 406 when a new customer's requirements are found to be infeasible. The process first to looks in step 502 for a reduced request submission rate for prospective customer j that is feasible. If the original request rate is λ_j, then the new request rate would be $\alpha\lambda_j$ for some α, $0<\alpha<1$. In order to have associated new UL and GM bounds that impose similar constraints with the new arrival rate, the UL and GM bounds are changed to keep $\alpha\lambda_j + c\sqrt{\alpha\lambda_j}$ constant, where c is the value associated with the case $\alpha=1$. The peakedness parameter, if any, is left unchanged. Since the search is only over the arrival rate, it is relatively elementary, i.e., binary search can be used. For each candidate rate $\alpha\lambda_j$, the blocking probability computer is used to check feasibility. If only a finite set of rates are available, then the search is over this set.

Next, in step 503, another feasible alternative is generated by keeping the initial rate fixed but changing the GM and UL bounds. First decrease the GM bound and see if a feasible solution can be obtained. If possible, find the maximum feasible GM bound by doing a search. If eliminating the GM bound does not produce a feasible solution, then set the GM bound equal to 0 and reduce the UL bound. Then do another search to find the maximum feasible UL bound. Then raise the GM bound to the minimum value of the previous GM bound and the highest value that does not affect feasibility. This scheme produces a new pair of UL and GM bounds with the same arrival rate that is feasible. The procedure to generate alternative grades of service stops in step 505.

The two alternatives generated in steps 502 and 503 are offered to the customer in step 310. The customer and resource provider might also wish to consider other alternatives that can be checked for feasibility as well. If a feasible alternative is acceptable to the customer, then the customer is admitted (step 309 of Fig. 3).

As indicated in step 303 of Fig. 3, providing multiple grades of service with UL and GM bounds requires that these bounds be enforced on an ongoing basis. The overall *procedure for efficiently enforcing the traffic bounds* is shown in Fig. 6.

Advantageously, this process checks the guaranteed-minimum bounds in an efficient manner. In particular, the computational complexity at each request event is of the same order as the number of resources used by the customer making that request This is much less complex than a scheme which requires computation complexity of the order equal to the product of the total number of resources and the total number of active classes, which might be quite large.

In Fig. 6, the first step 601 in enforcing the traffic bounds is initializing the control variables at startup of the system and when a customer arrives or departs. The customer and system parameters are:

p - number of resources

r - number of customers

K_i - capacity of resource i

L_j - guaranteed-minimum bound on the requests from customers j

U_j - upper-limit bound on the requests from customer j

b_{ij} - number of units in resource i required by each request of customer j

A key variable is the number n_j of active requests for customer j. Before any request arrivals, the value is $n_j = 0$.

Also, for efficient updating, another key variable is the number F_i of free units in resource i for each i. The initial value of F_i is the total number of units minus the number required by guaranteed minimum bounds, i.e.,

$$F_i = K_i - \sum_{j=1}^{r} L_j b_{ij} \ . \qquad (1)$$

The process of Fig. 6 waits at step 602 until a new request event occurs, and then proceeds to step 603. If it is determined in step 603 that a customer-j request completes service and departs, variables n_j and F_i are updated as follows:

(i) $n_j = n_j - 1$

(ii) For all i such that $b_{ij} > 0$, if $n_j \geq L_j$, then $F_i = F_i + b_{ij}$.
Otherwise do not change F_i.

If it is determined in step 603 that a new customer-j request arrived, the new customer-j request is admitted if it is determined in step 605 that the new request satisfied the UL and GM bounds, so that both

(i) $n_j \leq U_j - 1$ and

(ii) for each i such that $b_{ij} > 0$,

$$n_j b_{ij} \leq F_i + L_j b_{ij} - b_{ij}.$$

If a NO result occurs in step 605, the customer-j request is not admitted. If the customer-j request is admitted in step 605, the variables are updated in step 606 as follows:

(i) $n_j = n_j + 1$

(ii) For each i such that $b_{ij} > 0$, if $n_j > L_j$, then $F_i = F_i - b_{ij}$.

When the system starts with some initial number of requests, F_i must be computed for each i, using

$$F_i = K_i - \sum_{j=1}^{r} \max\{L_j b_{ij}, n_j b_{ij}\} \ . \qquad (2)$$

The overall computational complexity is O(rp). At each subsequent request arrival/departure event of a customer-j request, a computation which has computational complexity O(q) is performed where q is the number of resources used by customer j. In some applications, r and p will both be large, while q is still small. Accordingly, there may be a one-time large start-up computation, but at each request arrival/departure instant the amount of computation remains low and is of the same order as that for other simple request admission policies, such as complete-sharing and trunk reservation.

We now discuss ways to determine the *cost of providing service* to a new customer. There are several possible interpretations for cost of providing service to a new customer, even if we focus only on the capacity used on each resource. Clearly, the minimum capacity used is the guaranteed-minimum bound GM, and the maximum capacity used is the upper limit bound UL. It is natural to look for a notion of "expected capacity used", which should fall between these two extremes.

One expected cost expression is the *marginal expected cost,* which is the extra capacity required in a resource beyond what is required for all other customers using that resource, assuming that all customers submit requests according to their specified parameters. The marginal expected cost can be determined by first finding the minimum capacity of the resource, such that all current customer requirements can be met, and then finding the minimum capacity of the resource such that the requirements of all current customers plus the new customer can be met. The marginal expected cost of providing service to this customer on this resource is the difference between these two capacity levels. The blocking probability computer can be used to determine the two critical capacity levels, just as in the capacity adjustment procedure.

Another expected cost expression is the *first-customer expected cost,* which is the average capacity used if that customer were the only customer using the resource. This cost can be determined by finding the minimum capacity of the resource such that the customer's requirements can be satisfied, assuming that no other customers are present and that this customer submits requests according to its agreed-upon traffic parameters. The first-customer expected cost will typically be higher than the marginal expected cost, because there is less sharing. This first-customer expected

cost can also be determined by invoking the blocking probability computer, just as with the marginal expected cost. The calculation is easier with only a single customer.

It is also possible to use the normal approximation to obtain very rapid capacity requirement estimations for evaluating the two expected costs above. The normal approximation can be used as described below to estimate the required capacity in order to eliminate very lightly loaded resources from the model before performing numerical inversion.

We now discuss a treatment of *different kinds of arrival-process variability.* The usual assumption is that each customer's request arrival process can be reasonably modeled as a Poisson process. Then each request arrival process can be specified by simply giving the arrival rate. A Poisson process is often an appropriate model, but sometimes the request arrivals are in fact significantly more or less variable or "bursty" than in a Poisson process. For example, when requests are actually overflows of unsatisfied requests to another resource, they tend to arrive in a process that is more bursty than a Poisson process. A way to characterize non-Poisson processes is via the *peakedness parameter,* described by A. E. Eckberg, "Generalized Peakedness of Teletraffic Processes", Proceeding of the Tenth International Teletraffic Congress, Montreal, Canada, Paper 4.4b.3, and L. E. N. Delbrouck, "A Unified Approximate Evaluation of Congestion Functions for Smooth and Peaky Traffic", IEEE Transactions on Communications, volume 29, pages 85-91, 1981. "Peakedness" is defined as the ratio of the variance to the mean number of active requests in the associated infinite-capacity resource. For the case of Poisson arrivals, the number of requests in service has a Poisson distribution when there is no capacity limit. Since the variance equals the mean in a Poisson distribution, the peakedness is 1 for a Poisson arrival process. For a more bursty arrival process, the peakedness is greater than 1, while for a less bursty arrival process, the peakedness is less than 1.

Beginning with R. I. Wilkinson in "Theories for Toll Traffic Engineering in the U. S. A.", Bell System Technical Journal, volume 35, No. 2, pages 421-514, 1956, several researchers have made the observation that the number of active requests in a finite-capacity resource tends to have a truncated Pascal (or negative binomial) distribution for sources burstier than Poisson and a binomial distribution for sources smoother than Poisson, for any resource capacity. Exploiting this observation, L. E. N. Delbrouck in the reference cited earlier, developed accurate approximations for blocking computation in the context of single-rate, single-resource systems with the complete-sharing policy. We generalize that method in our more general setting.

The basic idea is to use a linear state-dependent arrival-rate function $\lambda_j(k)=\alpha_j+k\beta_j$, since this arrival process, known as a BPP process, produces a number of busy servers in an infinite-capacity resource distributed as Pascal for $\beta_j>0$ and as binomial for $\beta_j<0$. (Moreover, the BPP model falls within the scope of our blocking probability computer.) The parameters α_j and β_j of the BPP process are determined by matching the mean and variance of the number of class-j requests in service at an arbitrary time in an infinite-capacity resource, i.e., a system without any capacity constraints. Specifically, let M_j and V_j represent the mean and variance quantities mentioned above for the actual arrival process corresponding to customer j. It can be shown that these quantities for the approximating BPP process are

$$M_j = \frac{\alpha_j}{\mu_j - \beta_j} \tag{3}$$

and

$$V_j = \frac{\mu_j \alpha_j}{(\mu_j - \beta_j)^2}. \tag{4}$$

From the above, the two BPP parameters are

$$\alpha_j = M_j \mu_j / z_j \tag{5}$$

and

$$\beta_j = \mu_j (z_j - 1)/z_j, \tag{6}$$

where

$$z_j \equiv \text{peakedness} = V_j / M_j. \tag{7}$$

The quantity M_j is the offered load and is easily computable for most arrival processes. Also, for most common arrival processes, the peakedness parameter is readily computable as well, as indicated by A. E. Eckberg (cited above). These parameters can also be estimated from data.

Having obtained α_j and β_j, we wish to compute the blocking probabilities. This could be done directly by applying the blocking probability computer, but as noted by Delbrouck (cited earlier), a better approximation is obtained if we calculate the blocking probability indirectly via the mean number of active requests in the finite-capacity system with the BPP arrival process. Hence, the next step is to compute m_j, the mean number of class - j requests in service in the actual system with capacity constraints. It can be shown that m_j is given by

$$m_j = (\alpha_j/\mu_j) \frac{\bar{g}(K - Be_j n_j,\ U - e_j,\ N - Be_j n_j)}{g(K,M,N)}, \tag{8}$$

where e_j is a vector with a 1 in the j^{th} place and 0's elsewhere, and the symbol $\bar{g}$ in the numerator of Equation 8 implies that we have to consider a system with α_j replaced by $\alpha_j+\beta_j$. Note that this replacement is only for computing the numerator normalization constant and not the denominator one.

Hence m_j is readily computable by the blocking probability computer. Finally, the expression for call blocking is

$$B_j = 1 - \frac{m_j}{M_j} = 1 - (1 - \frac{\beta_j}{\mu_j}) \frac{\bar{g}(K-Be_j n_j, U-e_j, N-Be_j n_j)}{g(K,U,N)} \tag{9}$$

We now turn to the *blocking probability computer*, which is depicted in Fig. 2 and is invoked in step 307 of the admission control process in Fig. 3. (It plays a prominent role in Sections B and C below as well.) The blocking probability computer process depicted in step 307 of Fig. 3 is described in more detail in Fig. 7.

The process performed in the blocking probability computer requires the following resource-sharing inputs, which are obtained in step 701:

p - number of resources
r - number of customers
K_i - Capacity of resource i, $1 \le i \le p$
b_j - number of units required on each resource used by customers j

$$\delta_{ij} = \begin{cases} 1 & \text{if customer j uses resource i} \\ 0 & \text{otherwise} \end{cases}$$

B - matrix with $b_{ij} = \delta_{ij} b_j$
U_j - UL bound on requests from customer j, $1 \le j \le r$,
L_j - GM bound on requests from customer j
N_j - number of units guaranteed for customer j, $N_j = L_j b_j$
$K = (K_1,...,K_p)$ - capacity vector
$U = (U_1,...,U_r)$ - UL bound vector
$N = (N_1,...,N_r)$ - GM bound vector

If the model is too large to efficiently solve directly, a normal-approximation process is invoked in step 702 in order to produce an essentially equivalent smaller model. The approximate analysis looks for resources that are so lightly loaded that they impose no significant constraint, and a smaller model is made by eliminating these resources. This normal approximation scheme for eliminating lightly loaded resources is described in further detail below.

Next, in step 703, the generating function of the normalization constant associated with the product-form steady-state distribution is formed. First, the set of allowable states is

$$S(K,U,N) =$$

$$\{n \in Z_+^r : Bn \le K, n \le U, \sum_{j=1}^{r} (b_j n_j \wedge \delta_{ij} n_j) \le K_i, 1 \le i \le p\},$$

where Z_+^r is the set of *state vectors*, $n = (n_1,...,n_r)$ and n_j is the number of customer-j requests in service. The *steady-state distribution* is then

$$\pi(n) = g(K,U,N)^{-1} f(n), \tag{10}$$

where g(K,U,N) is the *normalization constant* and

$$f(n) = \prod_{j=1}^{r} f_j(n_j), \tag{11}$$

with

$$f_j(n_j) = \Lambda_j(n_j)/M_j(n_j),$$

$$\Lambda_j(n_j) = \prod_{k=0}^{n_j-1} \lambda_j(k),$$

$$M_j(n_j) = \prod_{k=1}^{n_j} \mu_j(k),$$

$\lambda_j(k)$ the arrival-rate function and $\mu_j(k)$ the service rate function. Then the normalization constant is

$$g(K,U,N) = \sum_{n \in S(K,U,N)} f(n) \ . \qquad (12)$$

The *generating fraction of the normalization constant* g(K,U,N) is

$$G(z,y,z) \equiv$$

$$\sum_{K_1=0}^{\infty} \dots \sum_{N_r=0}^{\infty} g(K,U,N) z_1^{K_1} \dots z_p^{K_p} y_1^{U_1} \dots y_r^{U_r} x_1^{N_1} \dots x_r^{N_r} \ , \qquad (13)$$

where $z=(z_1, \cdots, z_p)$, $y=(y_1, \cdots, y_r)$ and $x=(x_1, \cdots, x_r)$ are vectors of complex variables. The generating function in Equation 4 has the form

$$G(z,y,x) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} G_j(z,y,x) \ , \qquad (14)$$

where

$$G_j(z,y,x) = (1-y_j)^{-1}[(1-x_j \prod_{i=1}^{p} z_i^{\delta_{ij}})^{-1} F_j(y_j x_j^{b_j} \prod_{i=1}^{p} z_i^{b_{ij} b_j})$$

$$+(1-x_j)^{-1} F_j(y_j \prod_{i=1}^{p} z_i^{b_{ij}}) - (1-x_j)^{-1} F_j(y_j x_j^{b_j} \prod_{i=1}^{p} z_i^{b_{ij}})] \qquad (15)$$

and

$$F_j(x) \equiv \sum_{j=0}^{\infty} f_j(n_j) x^{n_j} \ , \qquad (16)$$

where x in Equation 16 is a single complex variable.

In the standard (Poisson) case with $\lambda_j(k) = \lambda_j$, $\mu_j(k) = k\mu_j$ and $\rho_j = \lambda_j/\mu_j$, the generating function $F_j(x)$ is given by:

$$F_j(x) = \exp(\rho_j x). \qquad (17)$$

In the so-called binomial and Pascal cases with $\lambda_j(k) = \alpha_j + \beta_j k$ where $\beta_j \neq 0$ and $r_j = \alpha_j/\beta_j$, and still $\mu_j(k) = k\mu_j$, the generating function $F_j(x)$ in Equation 16 is given by:

$$F_j(x) = (1-\beta_j x/\mu_j)^{-r_j}. \qquad (18)$$

For the special case in which $\lambda_j(k) = \lambda_j$ and $\mu_j(k) = \mu_j$, $f_j(n_j) = \rho_j^{n_j}$ the generating function $F_j(x)$, in Equation 16 is given by:

$$F_j(x) = (1-\rho_j x)^{-1}. \qquad (19)$$

The closed-form expressions for $F_j(x)$ in Equations 17 through 19 make it easier to calculate the generating function values in Equation 15, and thus in Equation 14. However, even in the general case, the infinite sum in Equation 16 can always be truncated without loss of generality. Because of the finite capacity limits, it suffices to assume that $\lambda_j(n_j) = 0$ for suitably large n_j, so that $f_j(k) = 0$ for all $k \geq n_j + 1$.

Even though some resources may have been eliminated from the process in step 702 in Fig. 7, the resulting process may still be too complex to solve directly. This difficulty is primarily due to the dimension of the generating function formed in step 703 being too large. When the dimension is too large, a good conditional decomposition is sought to reduce the effective dimension of the generating function. The procedure for finding a good conditional decomposition to achieve dimension reduction is described below.

Applying steps 702 and 704, a determination is made in step 705 as to whether blocking probabilities can be computed. If the result is "NO", a reduced-load fixed-point approximation is invoked in step 706. The blocking probability computer is actually used in this step too, as described in more detail below.

If the result in step 705 is "YES", the generating function is inverted in step 707. The inversion can be done in different ways. One effective inversion technique that exploits the Fourier-series method is described in J. Abate and W.Whitt, "The Fourier-Series Method for Inverting Transforms of Probability Distributions," Queueing Systems, volume 10, pages 5-88, 1992, and in G. L. Choudhury, D. M. Lucantoni and W. Whitt, "Multidimensional Transform Inversion With Applications to the Transient M/G1 queue," Annals of Applied probability, volume 4, pages 719-740, 1994.

Suppose that a p-dimensional generating function

$$G(z) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} g(K) z_1^{K_1} \cdots z_p^{K_p} \qquad (20)$$

is to be inverted. It is assumed that conditional decomposition has already been done to determine a good order in which the variables should be inverted. The procedure then is to perform up to p one-dimensional inversions recursively.

To represent the *recursive inversion,* let the *partial generating functions* be

$$g^{(j)}(z_j, K_{j+1}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_j=0}^{\infty} g(K) \prod_{i=1}^{j} z_i^{K_i} \text{ for } 0 \le j \le p, \qquad (21)$$

where $z_j = (z_1, z_2, \ldots, z_j)$ and $K_j = (K_j, K_{j+1}, \ldots, K_p)$ for $1 \le j \le p$. Let z_0 and K_{p+1} be null vectors. Clearly, $K = K_1$, $z = z_p$, $g^{(p)}(z_p, K_{p+1}) = G(z)$ and $g^{(0)}(z_0, K_1) = g(K)$.

Let I_j represent inversion with respect to z_j. Then the step-by-step nested inversion approach is

$$g^{(j-1)}(z_{j-1}, K_j) = I_j[g^{(j)}(z_j, K_{j+1})], \ 1 \le j \le p, \qquad (22)$$

starting with j = p and decreasing j by 1 each step. In the actual program implementation, the inversion in Equation 22 is attempted for j = 1. In order to compute the righthand side, another inversion with j = 2 is needed. This process goes on until at step p the function on the righthand side becomes the p-dimensional generating function and is explicitly computable.

Shown below is the inversion formula at the j^{th} step. For simplicity, those arguments which remain constant during this inversion are suppressed, letting $g_j(K_j) = g^{(j-1)}(z_{j-1}, K_j)$ and $G_j(z_j) = g^{(j)}(z_j, K_{j+1})$. With this notation, the inversion formula (Equation 22) is

$$g_j(K_j) = \frac{1}{2l_j K_j r_j^{K_j}} \sum_{k=-l_j K_j}^{l_j K_j - 1} G_j(r_j e^{\pi i k / l_j K_j}) e^{-\pi i k / l_j} - e_j, \qquad (23)$$

where $i = \sqrt{-1}$, l_j is a positive integer, r_j is a positive real number and e_j represents the aliasing error, which is given by

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2n l_j K_j) r_j^{2n l_j K_j}. \qquad (24)$$

Note that, for j = 1, $g_1(K_1) = g(K)$ is real, so that $G_1(\bar{z}_1) = \overline{G_1(z_1)}$. This enables the computation in Equation 23 to be reduced by about one half.

To control the aliasing error in (24), let $r_j = 10^{-a_j}$ for $a_j = \gamma_j/(2l_j K_j)$. Then Equation 24 becomes

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2n l_j K_j) 10^{-\gamma_j n}. \qquad (25)$$

A bigger γ_j in Equation 25 decreases the aliasing error. The parameter l_j controls roundoff error, with bigger values causing less roundoff error. An inner sum of the inversion requires more accuracy than an outer sum, since the inverted values in an inner sum are used as transform values in an outer sum. With a goal of about eight significant digit accuracy, the following sets of l_j and γ_j typically are adequate:i)$l_1 = 1$, $\gamma_1 = 11$, ii) $l_2 = l_3 = 2$, $\gamma_2 = \gamma_3 = 13$, iii)$l_4 = l_5 = l_6 = 3$, $\gamma_4 = \gamma_5 = \gamma_6 = 15$, assuming that computations are done using double-precision arithmetic. It is usually not a good idea to use the same l_j for all j, because then more computation is done to achieve the same accuracy.

When the inverse function is a probability, the aliasing error e_j in Equation 25 can easily be bounded because $g_j(K_j) \le 1$. In contrast, here the normalization constants may be arbitrarily large and therefore the aliasing error e_j may also be arbitrarily large. Thus, the generating function is scaled in each step by defining a *scaled generating function* as

$$\bar{G}_j(z_j) = \alpha_{0j} G_j(\alpha_j z_j), \qquad (26)$$

where α_{0j} and α_j are positive real numbers. This scaled generating function is inverted after choosing α_{0j} and α_j, so that the errors are suitably controlled. The inversion of $\bar{G}_j(z_j)$ in Equation 26 yields the *scaled normalization constant*

$$\bar{g}_j(K_j) = \alpha_{0j} \alpha_j^{K_j} g_j(K_j). \qquad (27)$$

An effective scaling procedure is described in more detail below. For large models, the computation can be speeded up by exploiting multiplicities, by judiciously truncating large finite sums, and by simultaneously computing many closely related normalization constants, with the bulk of the computation shared. These techniques are described in more

detail below. Finally, it remains to calculate the blocking probabilities themselves. This step is also performed in step 707, and is described in more detail below.

Turning now to the issue of *scaling*, it is noted that the aliasing error in the numerical inversion is controlled by scaling the generating function at each step. In the j^{th} step of a p-dimensional inversion, a scaled generating function is defined in Equation 26. This scaled generating function is inverted after choosing α_{0j} and α_j so that the errors are suitably controlled. The parameters α_{0j} and α_j in Equation 26 are chosen to control the *aliasing error with scaling*

$$\bar{e}_j = \sum_{n=1}^{\infty} \bar{g}_j(K_j + 2nl_jK_j)10^{-\gamma_j n} . \qquad (28)$$

Since the blocking probabilities involve ratios of normalization constants, it is appropriate to focus on relative errors $e_j' = \bar{e}_j / \bar{g}_j(K_j)$, which can be bounded by

$$|e_j'| \le \sum_{n=1}^{\infty} \left| \frac{\bar{g}_j(K_j + 2nl_jK_j)}{\bar{g}_j(K_j)} \right| 10^{-\gamma_j n} . \qquad (29)$$

Let

$$C_j = \max_n \left\{ \left| \frac{\bar{g}_j(K_j + 2nl_jK_j)}{\bar{g}_j(K_j)} \right| \right\}^{1/n}$$

$$= \alpha_j^{2l_jK_j} \max_n \left\{ \left| \frac{g_j(K_j + 2nl_jK_j)}{g_j(K_j)} \right| \right\}^{1/n} . \qquad (30)$$

Then

$$|e_j'| \le \sum_{n=1}^{\infty} C_j^n 10^{-\gamma_j n} \le \frac{C_j 10^{-\gamma_j}}{1 - C_j 10^{-\gamma_j}} \approx C_j 10^{-\gamma_j} . \qquad (31)$$

Note that C_j in Equation 30 is independent of α_{0j}. The second parameter α_{0j} is used mainly to keep $\bar{g}_j(K_j)$ close to 1, so as to avoid numerical underflow or overflow. (This numerical problem also can be avoided by working with logarithms.)

Hence, the main goal is to choose α_j so that $C_j << 10^n$, where << means "much smaller than." Of course, in general $g_j(K_j)$ is not known, so that C_j is not known as well. However, $C_j << 10^n$ can be achieved by roughly controlling the growth rate of $\bar{g}_j(K_j)$, or its fastest growing term, exploiting the structure of the generating function.

For the case of the complete sharing (CS) policy with Poisson arrivals, the scaled generating function is

$$\bar{G}(z_1, \ldots, z_p) = \frac{\prod_{i=1}^{p} \alpha_{0i} \exp\{ \sum_{j=1}^{r} p_j \prod_{i=1}^{p} (\alpha_i z_i)^{b_{ij}} \}}{\prod_{i=1}^{p} (1 - \alpha_i z_i)} \qquad (32)$$

and the scaled normalization constant is

$$\bar{g}(K) = \prod_{i=1}^{p} (\alpha_{0i} \alpha_i^{K_i}) g(K) . \qquad (33)$$

In this setting, an effective scaling is obtained by choosing the *scaling parameters* α_i, $1 \le i \le p$, so that they satisfy the inequalities $0 < \alpha_i \le 1$ and

$$\sum_{j=1}^{r} \rho_j \prod_{k=1}^{p} \alpha_k^{b_{kj}} \prod_{k=1}^{i-1} r_k^{b_{kj}} b_{ij} \le K_i , \quad 1 \le i \le p , \tag{34}$$

where $r_k = 10^{-aj}$ for $a_j = \gamma_j/(21_jK_j)$. Once the scaling variables α_i have been obtained, the variables α_{0i} are obtained recursively starting with i = p by

$$\prod_{k=i}^{p} \alpha_{0k} = \exp\left\{ -\sum_{j=1}^{r} \rho_j \prod_{k=1}^{p} \alpha_k^{b_{kj}} \prod_{k=1}^{i-1} r_k^{b_{kj}} \right\} . \tag{35}$$

A maximal vector $(\alpha_1, \ldots, \alpha_p)$ satisfying Equation 34 is found by starting with i = p and successively decreasing i. Use the fact that the left side of Equation 34 is monotone in α_i. When i = 1, the values of α_i for $i \ge 1 + 1$ are known. Approximate by acting as if $\alpha_i = 1$ for $i \le 1\text{-}1$ and find α_i satisfying the constraint for i = 1 in Equation 34. This yields a maximal vector, i.e., if any α_i is less than 1, then at least one constraint in Equation 34 is necessarily satisfied as an equality. Hence, the vector cannot be increased without violating a constraint. However, in general there may be many maximal vectors.

Now the scaling for other arrival-rate and service-rate functions will be described. Only the one-dimensional case will be described in detail. The multidimensional extensions parallel Equations 34 and 35 above.

First, consider the case in which $\lambda_j(k) = \alpha_j + k\beta_j$ for $\beta_j \ne 0$ and $\mu_j(k) = k\mu_j$. (The complete sharing policy is still assumed at this point.) In the one-dimensional case,

$$\begin{aligned} (1-z)G(z) &= \prod_{j=1}^{r} (1-(\beta_j/\mu_j)z^{b_j})^{-r_j} \\ &= \prod_{j=1}^{r} \exp(-r_j \ln(1-(\beta_j/\mu_j)z^{b_j})) \\ &= \prod_{j=1}^{r} \exp\Big(\sum_{l=1}^{\infty} (\beta_j/\mu_j)^l (r_j/l) z^{lb_j}\Big) \\ &= \prod_{j=1}^{r} \prod_{l=1}^{\infty} \exp\left[\left[(\beta_j/\mu_j)\right]^l (r_j/l) z^{lb_j} \right] . \end{aligned} \tag{36}$$

From Equation 36, it follows that this case can be regarded as the Poisson case with infinitely many classes. The classes are indexed as (j,1) where $1 \le j \le r$ and $1 \le 1 \le \infty$. The traffic intensity parameter for class (j,1) is $(\beta_j/\mu_j)^1(r_j/1)$ and the resource units needed by a member of this class in $1b_j$. Therefore the same scaling can be used as in the Poisson case.

The one-dimensional version of Equation 34 becomes

$$\sum_{j=1}^{r} \frac{r_j(\beta_j/\mu_j)\alpha^{b_j} b_j}{1-(\beta_j/\mu_j)\alpha^{b_j}} \le K . \tag{37}$$

Thus, α is the largest number in (0,1] satisfying Equation 37. When solving Equation 37, the left side should be taken as infinity if any of the denominator terms are negative.

Next

$$\alpha_0 = \exp\Big(-\sum_{j=1}^{r} \sum_{l=1}^{\infty} (\beta_j/\mu_j)^l (r_j/l)\alpha^{lb_j}\Big) \tag{38}$$

and, after simplification,

$$\alpha_0 = \prod_{j=1}^{r} (1-(\beta_j/\mu_j)\alpha^{b_j})^{r_j} . \tag{39}$$

Next consider the case with $\lambda_j(k) = \lambda_j$ and $\mu_j(k) = \mu_j$. This corresponds to a buffered model with a single server, as in F. Kamoun and L. Kleinrock "Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions", IEEE Transactions on Communications, volume COM-28, pages 992-1003, 1980. This case can be reduced to the case just considered by replacing r_j by 1 and (β_j/μ_j) by ρ_j. That yields the scaling parameters. Specifically, α is the maximal solution in the range (0,1] of the equation

$$\sum_{j=1}^{r} \frac{\rho_j \alpha^{b_j} b_j}{1-\rho_j \alpha^{b_j}} \leq K . \tag{40}$$

As before, if $1-\rho_j\alpha^{b_j} < 0$ for some j, the entire left hand side is to be interpreted as infinity. Then

$$\alpha_0 = \prod_{j=1}^{r} (1-\rho_j \alpha^{b_j}) . \tag{41}$$

So far, the complete-sharing (CS) policy has been assumed. In the UL case, the form of the generating function is the same as a CS generating function with more resources. In this case, the CS scaling can be extended in a straightforward way. The GM case can be treated heuristically (only for scaling) by equating it to a UL policy with the upper limits equal to the capacities minus the sum of the guaranteed minima of all other classes. For the combined UL and GM case, as an approximation, use new upper limits, which for each class are the minimum of the given upper limit and the heuristic one based on the guaranteed minima of all other classes.

There is another heuristic that applies to generating functions of any form, and thus is applicable to the general state-dependent arrival and service rates. Let,

$$G(z) = (1-z)^{-1} \prod_{j=1}^{r} G_j(z) \tag{42}$$

with

$$G_j(z) = \sum_{n_j=0}^{\infty} f_j(n_j)(z^{b_j})^{n_j} . \tag{43}$$

Then α is the largest number in the interval (0,1] such that

$$\sum_{j=1}^{r} \frac{zG_j'(z)}{G_j(z)}\Big|_{z=\alpha} \leq K , \tag{44}$$

where $G_j'(z) = \frac{d}{dz} G_j(z)$, i.e., the derivative.

Then

$$\alpha_0^{-1} = \prod_{j=1}^{r} G_j(\alpha) . \tag{45}$$

This scaling agrees with the previous scalings in the special cases treated in detail.

The p-dimensional extension of Equation 42 is

$$G(z) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} G_j(z), \tag{46}$$

where,

$$G_j(z) = \sum_{n_j=0}^{\infty} f_j(n_j) \prod_{i=1}^{p} z_i^{b_{ij} n_j} . \tag{47}$$

It is assumed that the innermost level of inversion is with respect to z_1 and outermost level of inversion is with respect to z_p. Let the scaling parameters be $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_p)$ and $\alpha_0 = (\alpha_{01}, \alpha_{02}, \ldots, \alpha_{0p})$. Then the scale vector α should be a

maximal vector satisfying $0 < \alpha_i \leq 1$ for i = 1,2,..., p and

$$\sum_{j=1}^{r} \left(\prod_{k=1}^{i-1} r_k^{a_{kj}}\right) z_i \frac{\partial}{\partial z_i} \ln G_j(z)|_{z=\alpha} \leq K_i \qquad (48)$$

for i = 1,2,...p, where

$$r_k = 10^{-\gamma_k/21_k K_k} . \qquad (49)$$

By "maximal" is meant that Equation 48 should be satisfied with equality for at least one i unless $\alpha = (1,1,\ldots, 1)$. The remaining scale parameters α_{0i}, $1 \leq i \leq p$, are obtained recursively starting with i = p by

$$\prod_{k=i}^{p} \alpha_{0k} = \left[\prod_{j=1}^{r} G_j(\alpha_1 r_1, \alpha_2 r_2, \ldots, \alpha_{i-1} r_{i-1}, \alpha_i, \alpha_{i+1}, \ldots, \alpha_p)\right]^{-1} . \qquad (50)$$

Turning now to several techniques that can advantageously be used as options in connection with the present invention, it is to be noted first that if two or more classes have the same parameters (traffic parameters, resource requirements, UL and GM parameters), then this multiplicity can be exploited to significantly reduce the required computation.

Let $\bar{r}$ be the number of different *types of customers,* and let the j^{th} type have *multiplicity* m_j. Then the total number of customer classes is

$$r = \sum_{j=1}^{\bar{r}} m_j . \qquad (51)$$

If the p-dimensional generating function of interest can be written as

$$G(z) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} G_j(z) , \qquad (52)$$

then it can be rewritten as

$$G(z) = \prod_{i=1}^{p} (1-z_i) \prod_{j=1}^{\bar{r}} G_j(z)^{m_j} . \qquad (53)$$

The computational complexity in evaluating Equation 52 is O(r), while the computational complexity in evaluating Equation 53 is $O(\bar{r})$.

Second, as can be seen from Equation 23, the inversion formula in each dimension is a sum of 21_iK_i terms. If K_i is large, there are ways to accelerate convergence of the finite sum including judicious truncation of the finite sum.

The inversion formula in each dimension is a weighted sum of generating function values evaluated over equidistant points along the circumference of a circle. The weights are complex numbers, but they have constant amplitude. As the capacities K_i grow, the amplitude of the generating function typically becomes unevenly distributed along the circumference of the circle. There are several local maximum points and the amplitude drops sharply away from these points. (Since the weights have constant amplitude, it is only necessary to consider the relative amplitude of the generating function values.) If all the relative maximum points can be identified, and then only those points around them that have non-negligible relative amplitude are considered, it is possible to obtain a significant reduction in computation.

First, the truncation procedure is developed for a single resource and then it is extended. Consider the scaled generating function in the case of complete sharing with Poisson arrivals, i.e.,

$$\bar{G}(z) = \alpha_0 \exp(\rho \alpha^b z^b)/(1-\alpha z). \qquad (54)$$

In the outer dimension the computation can be cut in half; thus consider the sum over the upper semicircle with radius $r_1 = 10^{-\gamma_1/21_1K_1}$.

At a summation point, $z_1 = r_1 e^{i\theta}$ where θ assumes the values $\pi k/(1_1K_1)$ for $0 \leq k \leq 1_1K_1$. Let $\bar{G}^*(\theta)$ be $\bar{G}(z_1)$ expressed as a function of θ, i.e.,

$$\bar{G}^*(\theta) = \frac{\alpha_{01} \prod_{j=1}^{r} \exp(\rho_j \alpha_1^{b_j} r_1^{b_j} e^{ia_j\theta})}{1-\alpha_1 r_1 e^{i\theta}} . \quad (55)$$

Note that the amplitude of $\bar{G}^*(\theta)$ in Equation 55 is

$$|\bar{G}^*(\theta)| = \frac{\alpha_{01} \prod_{j=1}^{r} \exp(\rho_j (\alpha_1 r_1)^{b_j} \cos(b_j\theta))}{\sqrt{1+\alpha_1^2 r_1^2 - 2\alpha_1 r_1 \cos\theta}} . \quad (56)$$

For j = 1,2,...,r, the numerator has relative maxima at $\theta = 21_j\pi/b_j$ for $1_j = 0,1,...,$

$$\left\lfloor b_j/2 \right\rfloor .$$

The denominator has a single minimum at θ = 0. Hence, $|\bar{G}^*(\theta)|$ has a *global maximum* at θ = 0 and *potential local maxima* at $\theta = 21_j\pi/b_j$ for $1_j = 1,2,...,$

$$\left\lfloor b_j/2 \right\rfloor$$

and j = 1,2,...,r. Note that usually many of these

$$\sum_{j=1}^{r} \left\lfloor b_j/2 \right\rfloor$$

local maxima will coincide.

In summary, start by computing $|\bar{G}^*(0)|$and then find the distinct local maximum points $\theta = 21_j\pi/b_j$ for $1_j = 1,...,$

$$\left\lfloor b_j/2 \right\rfloor$$

and j = 1,...r, and sort them in increasing order. Let these points be θ_m^i for i = 1,2,...,L. In general θ_m^i may not coincide with a summation point in the inversion algorithm. In that case, move θ_m^i to the nearest summation point used in the inversion algorithm. Next find all i such that $|G^*(\theta_m^i)/\bar{G}^*(0)| \geq \varepsilon$, where ε is some allowable error. Then

$$|\bar{G}^*(\theta)/\bar{G}^*(0)| =$$

$$\exp(-\sum_{j=1}^{r} \rho_j (\alpha_1 r_1)^{b_j} (1-\cos b_j\theta)) \frac{1-\alpha_1 r_1}{\sqrt{1+\alpha_1^2 r_1^2 - 2\alpha_1 r_1 \cos\theta}} . \quad (57)$$

For all these i, sum over all summation points in the inversion algorithm above and below θ_m^i until $|\bar{G}^*(\theta)/\bar{G}^*(0)| \geq \varepsilon$. Do not sum over any summation point more than once.

For large ρ_j and K_1, typically only the points around θ = 0 and a few other local maxima will be significant. How much computational savings will result can be seen by computing for all values of θ in $(0,\pi)$ and finding the proportion of the range for which $|\bar{G}^*(\theta)/\bar{G}^*(0)| > \varepsilon$. The savings has been shown to be approximately proportional to $\sqrt{K_1}$.

Truncation can also be exploited with multiple resources, but the situation is more complicated. Now the scaled generating function is given by

$$\bar{G}(z_1,\ldots,z_p) = \frac{\prod_{i=1}^{p} \alpha_{0i} \exp\left[\sum_{j=1}^{r} \rho_j \prod_{i=1}^{p} (\alpha_i z_i)^{b_{ij}}\right]}{\prod_{i=1}^{p} (1-\alpha_i z_i)} \quad (58)$$

and the scaled normalization constant is

$$\bar{g}(K) = \prod_{i=1}^{p} (\alpha_{0i}\alpha_i^{K_1}) g(K) \ . \qquad (59)$$

For inversion with respect to z_p, the same computational saving as for a single resource can be achieved, but there are two differences: First, the inversion formula involves summation over the entire circle instead of just a semicircle, so that it is necessary to consider more maximum points. Second, the constant $\rho_j\alpha_1^{b_j}$ in Equation 55 has to be replaced by the constant

$$\rho_j \prod_{l=1}^{p} \alpha_l^{b_{lj}} \prod_{l=1}^{p-1} z_l^{b_{lj}}.$$

Since the latter constant is a complex number, it will introduce a constant phase change to θ and hence to all maximum points.

For inversion with respect to z_i for $i < p$, it is necessary to cope with the partially inverted generating function $g^{(i-1)}(z_{i-1},K_j)$ for which the functional form is not known. Hence, the maximum points are not known, so that it is necessary to resort to heuristics. Assuming that the location of the maximum points is the same as if the partially inverted generating function has the same functional form as in Equation 55 usually works and gives good computational savings.

In order to compute the blocking probability for each of the r classes in step 706 and 707 of Fig. 7, the computational complexity is $O(r^2)$, because r + 1 normalization constant values have to be calculated in step 702 and the computation required for each is O(r). However, for large capacity vectors K it is possible to compute the r + 1 normalization constants simultaneously with the bulk of the computations shared, so that the required computation for all normalization constants is only slightly more than for one. This reduces the overall computational complexity from $O(r^2)$ to O(r).

To explain the method, consider a single resource with Poisson arrivals and the complete sharing policy. The blocking probability for class j is $B_j = 1\text{-}g(K_1\text{-}b_j)/g(K_1)$. Then, letting $b_0 \equiv 0$, it is necessary to compute $g(K_1 - b_j)$ for $0 \le j \le r$. Combining the scaling and inversion procedures already described, the standard formula for this computation is

$$g(K_1 - b_j) =$$

$$\frac{\alpha_{01j}^{-1}\alpha_{1j}^{-(K_1-b_j)}}{m_{1j}r_{1j}^{K_1-b_j}} \sum_{k=-\frac{m_{1j}}{2}}^{\frac{m_{1j}}{2}-1} \alpha_{01j} G(\alpha_{1j} r_{1j} e^{\frac{2\pi ik}{m_{1j}}}) e^{-\frac{2\pi ik(K_1-b_j)}{m_{1j}}} \ , \qquad (60)$$

where α_{01j} and α_{1j} are the scaling parameters,

$$m_{1j} = 21_1(K_1 - b_j) \text{ and } r_{1j} = 10^{-\gamma_1/m_{1j}}. \qquad (61)$$

The associated aliasing errors are

$$e_{1j} = \sum_{n=1}^{\infty} \alpha_{01j}\alpha_{1j}^{K_1-b_j} g(K_1 - b_j + nm_{1j}) 10^{-n\gamma_1} \ . \qquad (62)$$

Note that the computation in Equation 60 for different values of j cannot be shared because the quantities α_{01j}, α_{1j} and m_{1j} are different for different values of j.

In order to share computation for different values of j, for $K_1 >> b_j$ $(0 \le j \le r)$, replace the quantities α_{01j}, α_{1j} and m_{1j} by their values at j = 0 for all j. This should not cause any appreciable difference in the error expression in Equation 60 since for $K_1 >> b_j$ the quantities α_{01j}, α_{1j} and m_{1j} are pretty close to their values at j = 0 anyway.

Dropping the subscript j for α_{01j}, α_{1j} and m_{1j}, Equation 60 can be rewritten as

$$g(K_1 - b_j) = \frac{1}{m_1\alpha_{01}(\alpha_1 r_1)^{K_1-b_j}} \sum_{k=-m_1/2}^{m_1/2} T_k C_j^k \ , \qquad (63)$$

where

$$C_j = e^{\frac{2\pi ib_j}{m_1}} \text{ and } T_k = \alpha_{01} G(\alpha_1 r_1 e^{\frac{2\pi ik}{m_1}}) e^{\frac{2\pi ikK_1}{m_1}} \ . \qquad (64)$$

Note that the bulk of the computation in Equation 63 is computing T_k, which can be shared. The quantities C_j^k can be computed quickly, since C_j needs to be computed only once for each j. It is also clear that by working with partial sums for all j simultaneously, the overall computation may be done with a storage requirement O(r). Moreover, if truncation (described above) applies, then it applies uniformly for all j, since $|C_j^k| = 1$. For multiple resources and other sharing policies, the same approach works.

We now turn to a discussion of *computing the blocking probabilities.* The blocking probabilities are obtained as relatively simple expressions of the normalization constants, so that the main work is computing the normalization constants, which has been described above. Nevertheless, some care is needed in computing the blocking probabilities. It is important to distinguish between "call blocking" and "time blocking". "Call blocking" refers to the blocking experienced by arrivals (which depends on the state at arrival epochs), while "time blocking" refers to the blocking that would take place at an arbitrary time if there were an arrival at that time. Since the steady-state distribution π in Equation 10 refers to an arbitrary time, blocking probabilities computed directly from it involve time blocking, but it is possible to calculate call blocking as well as time blocking. With Poisson arrivals, the two probability distributions at arrival epochs and at an arbitrary time agree, but not more generally.

First, when we obtain a BPP state-dependent arrival process as an approximation to an arrival process partially specified by rate and peakedness, then (as discussed in the treatment of peakedness above) it is usually better to compute the call blocking probability using Equation 9 instead of computing the exact BPP call blocking. Hence, the discussion about call blocking below is intended for the case in which the state-dependent arrival process arises in the model naturally (as opposed to an approximate representation of peakedness). An important example is the case of finite-source input.

With the complete-sharing policy, the probability that a customer-j request would not be admitted at an arbitrary time (time blocking) is

$$B_j^t = 1 - \frac{g(K - b_j)}{g(K)}, \tag{65}$$

where $b_j \equiv (b_{1j}, \ldots, b_{pj})$ is the requirements vector for class j.

As noted above, if customer-j requests arrive in a Poisson process, then Equation 65 also yields the call blocking, but not more generally. However, if the arrival rate is state-dependent, then the call blocking always can be obtained by calculating the time blocking in a modified model. Let B_j be the customer-j blocking probability (call blocking). Let $B \equiv (b_{ij})$ be the requirements matrix. In general,

$$\begin{aligned} B_j &= 1 - \frac{\sum_{n:Bn \le K - b_j} \lambda_j(n_j)\pi(n)}{\sum_{n:Bn \le K} \lambda_j(n_j)\pi(n)} \\ &= 1 - \frac{\sum_{n:Bn \le K - b_j} \lambda_j(n_j) f(n)}{\sum_{n:Bn \le K} \lambda_j(n_j) f(n)} . \end{aligned} \tag{66}$$

However, $\lambda_j(n_j)f(n)$ can be rewritten as $\lambda_j(0)\dot{f}(n)$ and thus Equation 65 can be rewritten as

$$B_j = 1 - \frac{\sum_{n:Bn \le K - b_j} \dot{f}(n)}{\sum_{n:Bn \le K} \dot{f}(n)} = 1 - \frac{\dot{g}(K - b_j)}{\dot{g}(K)}, \tag{67}$$

where $\dot{f}(n)$ is the analog of f(n) with λ_j (m replaced by $\bar{\lambda}_j(m) \equiv \lambda_j(m+1)$, and $\dot{g}(K)$ is the analog of g(K) with f(n) replaced by $\dot{f}(n)$. Thus, the customer-j blocking probabilities B_j in Equation 67 coincide with the time-blocking quantities B_j^t in Equation 65 for the modified model in which the class-j arrival-rate function is changed from $\lambda_j(m)$ to $\bar{\lambda}_j(m) \equiv \lambda_j(m+1)$.

For the special case in which $\lambda_j(m) = \alpha_j + \beta_j m$,

$$\bar{\lambda}_j(m) = \lambda_j(m+1) = \alpha_j + \beta_j(m+1) = (\alpha_j + \beta_j) + \beta_j m, \tag{68}$$

so that the modified model is a model of the same general form. For the model with linear arrival-rate function, this approach to computing call blocking was pointed out by Z. Dziong and J. W. Roberts, in "Congestion Probabilities in a Circuit-Switched Integrated Services Network", Performance Evaluations, volume 7, pages 267-284, 1987, at page

273.

Different expressions are needed for the model with UL and GM bounds. Paralleling Equation 65, the time blocking with the UL and GM bounds is

$$B_j^t = 1 - \frac{g(K-Be_j, U-e_j, N-Be_j)}{g(K,U,N)}. \quad (69)$$

where g(K,U,N) is the normalization constant. The call blocking is again the time blocking B_j^t with modified parameters.

We now return to other key steps in the blocking probability computer, depicted in Fig. 7. As indicated in Fig. 7, step 702 of the blocking probability computer process is to apply the *normal-approximation algorithm to identify and eliminate very lightly loaded resources from the resource model.* This step is done before performing the inversion in step 707 to make the inversion less difficult.

The details of the normal approximation method are illustrated in Fig. 8. First, in step 801 the mean and variance of the capacity used by each customer on each resource is determined. For further discussion, let the resource and the customer be fixed, and omit the i and j subscripts.

It is assumed that the arrival-rate function is linear, i.e., $\lambda(k) = \alpha + k\beta$. As indicated above, this includes the standard case of Poisson arrivals and the case in which there is a peakedness parameter. If there were no capacity constraints, then the mean and variance of the number of requests in service would be

$$m = \frac{\alpha}{\mu - \beta} \text{ and } v = \frac{\mu\alpha}{(\mu-\beta)^2}. \quad (70)$$

Since each request uses b resource units, the associated mean and variance of the number of resource units used are

$$\bar{m} = mb \text{ and } \bar{v} = vb^2 \quad (71)$$

for m and v in Equation 70.

However, it remains to take account of the UL bound U and the GM bound L. For this purpose, a conditional normal approximation is used. The idea is to act as if the number of active requests at any time is a normally distributed random variable with mean m and variance v in Equation 70, but conditional on never being above U. This is denoted by $\bar{N} \equiv (N(m,v)|(N(m,v) \le U)$.

Let the capacity (number of resource units) used be denoted by C(L,U). Note that C(L,U) is bL when $\bar{N} \le L$, and is $b\bar{N}$ otherwise. Hence, using properties of the normal distribution, the mean and variance of C(L,U) can be calculated. For this purpose, let $\phi(x)$ be the standard (mean 0, variance 1) normal density function and let $\Phi(x)$ be its cumulative distribution function.

Assuming that the occupancy for a customer can be approximated by the conditional normal variable $(N(m,\sigma^2)|N(m,\sigma^2) \le U)$, the first two moments of the capacity used at any time are

$$\begin{aligned} EC(L,U) &= bLP(N(m,\sigma^2) \le L | N(m,\sigma^2) \le U) \\ &\quad + bE(N(m,\sigma^2) | L \le N(m,\sigma^2) \le U) P(L \le N(m,\sigma^2) | N(m,\sigma^2) \le U) \\ &= bL \frac{\Phi((L-m)/\sigma)}{\Phi((U-m)/\sigma)} + bX \left[\frac{\Phi((U-m)/\sigma) - \Phi((L-m)/\sigma)}{\Phi((U-m)/\sigma)} \right], \end{aligned} \quad (72)$$

where

$$X \equiv m + \sigma \frac{\phi((L-m)/\sigma) - \phi((U-m)/\sigma)}{\Phi((U-m)/\sigma) - \Phi((L-m)/\sigma)} \quad (73)$$

and

$$\begin{aligned} E[C(L,U)^2] &= b^2L^2P(N(m,\sigma^2) \le L | N(m,\sigma^2) \le U) \\ &\quad + b^2E[N(m,\sigma^2)^2 | L \le N(m,\sigma^2) \le U) P(L \le N(m,\sigma^2) | N(m,\sigma^2) \le U) \\ &= b^2L^2 \frac{\Phi((L-m)/\sigma)}{\Phi((U-m)/\sigma)} + b^2Y \left\{ \frac{\Phi((U-m)/\sigma) - \Phi((L-m)/\sigma)}{\Phi((U-m)/\sigma)} \right], \end{aligned} \quad (74)$$

where

$$Y = m^2 + \sigma^2 + 2m\sigma\left[\frac{\phi((L-m)/\sigma)-\Phi((U-m)/\sigma)}{\Phi((U-m)/\sigma)-\Phi((L-m)/\sigma)}\right]$$

$$+\sigma\left[\frac{(L-m)\phi((L-m)/\sigma)-(U-m)\phi((U-m)/\sigma)}{\Phi((U-m)/\sigma)-\Phi((L-m)/\sigma)}\right]. \quad (75)$$

As usual, the variance is

$$\text{Var } C(L,U) = E[C(L,U)^2] - E[C(L,U)]^2. \quad (76)$$

If, in addition, $\Phi((U-m)\sigma) \approx 1$ and $\phi((U-m)\sigma) \approx 0$, then

$$EC(L,U) \approx bm - b(m-L)\Phi((L-m)\sigma) + b\sigma\phi((L-m)/\sigma) \quad (77)$$

and

$$E[C(L,U)^2] \approx b^2L^2\Phi((L-m)\sigma) + b^2(m^2+\sigma^2)\Phi^c((L-m)\sigma) + b^2[2m\sigma + (L-m)]\phi((L-m)/\sigma)). \quad (78)$$

The description so far has shown how to estimate the mean and variance of the resource units used by a given customer. For many customers, let M_j and V_j denote the mean and variance of C(L,U) for customer j. The next step, step 802, is to add the means and variances over all customers to find a total mean and variance for all customers, say M and V (for one fixed resource).

The normal approximation is next used in step 803 to estimate the capacity actually needed. In particular, the total capacity needed at any time is regarded as approximately normally distributed, i.e., as N(M,V). If X is the capacity needed, then $(X - M)/\sqrt{V}$ is thus distributed as N(0,1). Thus, it can be determined if the available capacity K on this resource provides a serious constraint. For this purpose, construct the "binding parameter"

$$\gamma = (K-M)/\sqrt{V}. \quad (79)$$

Note that

$$P(X > K) \approx P(N(0,1) > \gamma). \quad (80)$$

Hence, if the binding parameter γ in Equation 79 is suitably large, e.g., when $\gamma \geq 5$, then the capacity K is unlikely to be exceeded.

The calculation just described is done for every resource. All resources that have suitably large binding parameters in Equation 79 are eliminated from the model, in step 804.

As indicated in Fig. 7, step 704 of the blocking probability computer involves applying a *conditional decomposition procedure to reduce the effective dimension of the transform.* As with the normal-approximation algorithm just discussed, this step is done before performing the inversions in step 707 to make the inversion less difficult. The idea behind conditional decomposition is that it is often possible to reduce the effective dimension of a numerical inversion by exploiting special structure. To understand this concept, note that if G(z) can be written as the product of factors, where no two factors have common variables, then the inversion of G(z) can be carried out by inverting the factors separately, and the dimension of the inversion is reduced to the maximum dimension of any one factor. The factors can be treated separately, because factors not involving the variable of integration pass through the sum in the inversion formula

However, it virtually never happens that the generating function can be factored into separate components with no common variables, because this corresponds to having two unrelated problems. The idea is to look for a weaker property, called "conditional decomposition". First, select d variables that will be inverted and see if the remaining function of p - d variables can be expressed as a product of factors with no two factors containing any common variables from these p - d remaining variables. The maximum dimension of the inversion is then d plus the maximum number of the p - d variables appearing in any one factor, say m. The overall inversion can then be regarded as being of dimension d+m. The idea is to select an appropriate d variables, so that the resulting dimension d+m is small. In small problems, this can be done by enumeration.

It is significant that conditional decomposition can always be systematically exploited for UL and GM policies to drastically reduce the effective dimension. Consider the generating function displayed in Equation 14. The generating function there is directly expressed in terms of factors. Since the factor $G_j(z,y,x)$ in Equation 15 contains only the variables z, y_j and x_j, the effective dimension in Equation 14. can always be reduced from p + 2r to p + 2. However, it is possible to do even better by explicit inversion. Explicit inversion with respect to x_j yields

$$G_j(z,y_j,N_j) = (1-y_j)^{-1}\left[\left(\prod_{i=1}^{p} z_i^{\delta_{ij}}\right)^{N_j} \sum_{n_j=0}^{\lfloor N_j/b_j \rfloor} f_j(n_j) y_j^{n_j} + \right.$$

$$\left. \sum_{n_j=\lfloor N_j/b_j \rfloor+1}^{\infty} f_j(n_j)\left(y_j \prod_{i=1}^{p} z_i^{a_{ij}}\right)^{n_j}\right]. \qquad (81)$$

Now doing explicit inversion of Equation 81 with respect to y_j and remembering that

$$U_j \geq \lfloor N_j/b_j \rfloor$$

yields

$$G_j(z,U_j,N_j) = \left(\prod_{i=1}^{p} z_i^{\delta_{ij}}\right)^{N_j} \sum_{n_j=0}^{\lfloor N_j/b_j \rfloor} f_j(n_j)$$

$$+ \sum_{n_j=\lfloor N_j/b_j \rfloor+1}^{U_j} f_j(n_j)\left(\prod_{i=1}^{p} z_i^{\delta_{ij}}\right)^{n_j b_j}. \qquad (82)$$

Note the remarkably simple form of Equation 82. Assuming N_j to be an integral multiple of b_j, rewrite Equation 82 as

$$G_j(z,U_j,N_j) = \left(\prod_{i=1}^{p} z_i^{\delta_{ij}}\right)^{N_j}$$

$$\times\left[\sum_{l=0}^{\lfloor N_j/b_j \rfloor} f_j(l) + \sum_{l=1}^{U_j-\lfloor N_j/b_j \rfloor} f_j\left(\lfloor N_j/b_j \rfloor+l\right)\left(\prod_{i=1}^{p} z_i^{\delta_{ij}}\right)^{lb_j}\right]. \qquad (83)$$

The overall remaining generating function is

$$G(z,U,N) = \prod_{i=1}^{p}(1-z_i)^{-1} \prod_{j=1}^{r} G_j(z,U_j,N_j), \qquad (84)$$

where $G_j(z,U_j,N_j)$ is given by Equation 81 or Equation 82. If Equation 82 is used, then there is a leading term

$$\prod_{i=1}^{p} z_i^{\sum_{j=1}^{r} N_j \delta_{ij}}$$

which can be explicitly taken out, yielding a smaller problem with K_i replaced by

$$K_i - \sum_{j=1}^{r} \delta_{ij} N_j$$

for i=1,2,...p. This step will be especially effective if

$$K \approx \sum_{j=1}^{r} \delta_{ij} N_j.$$

As an extreme case, if

$$K_i = \sum_{j=1}^{r} \delta_{ij} N_j,$$

$1 \le i \le p$, then this yields complete partitioning. Then Equations 82 and 83 provide an explicit expression for the normalization constant as

$$g(K,U,N) = \prod_{j=1}^{r} \sum_{n_j=0}^{\lfloor N_j/b_j \rfloor} f_j(n_j) . \qquad (85)$$

Using Equations 84 and 81 or Equation 83, the effective dimension of inversion has been effectively reduced from $p + 2r$ to p. However, since there are U_j terms in Equations 82 and 83, the computational complexity is about U_j times that of a closed form p-dimensional inversion. In general, U_j could increase with the K_i, but if there are many classes, U_j may remain small even with large K_i. If, however, U_j is indeed very large, then it will be advantageous to work with $G_j(z,y_j,N_j)$ and do one more level of inversion. If

$$\lfloor N_j/b_j \rfloor$$

is large, then it may even be advantageous to work with $G_j(z,y_j,x_j)$.

Special cases are easily obtained by inserting the corresponding expressions for $f_j(n_j)$. In case $\lambda_j(k) = \lambda_j$ and $\mu_j(k) = \mu_j$, the sums in Equations 82 and 83 may be expressed in closed form. Specifically, Equation 83 becomes

$$G_j(z,M_j,N_j) = \left[\prod_{i=1}^{p} z_i^{\delta_{ij}}\right]^{N_j} \left[\frac{1-\rho_j^{\lfloor N_j/b_j \rfloor + 1}}{1-\rho_j} + \left[1-\rho_j\left[\prod_{i=1}^{p} z_i^{\delta_{ij}}\right]^{b_j}\right]^{-1}\right] \times \left[\rho_j^{\lfloor N_j/b_j \rfloor + 1}\left[\prod_{i=1}^{p} z_i^{\delta_{ij}}\right]^{b_j} - \rho_j^{U_j+1}\left[\prod_{i=1}^{p} z_i^{\delta_{ij}}\right]^{U_j b_j - N_j + b_j}\right] . \qquad (86)$$

Thus, in this case the computation for the combined UL and GM model is just as fast as for the simple CS model.

The *reduced-load fixed-point approximation* performed in step 706 of Fig. 7 is an effective approach for approximately analyzing resource-sharing models or loss networks that are too large to solve exactly. This approach is also important for approximately analyzing systems in which there is *alternative routing* for blocked requests. In this approach, we act as if the different resources are stochastically independent, and the customer request arrival processes to different resources are mutually independent, but reduce the offered load of each customer at each resource by the blocking experienced by that customer elsewhere. This approximation strategy, together with a process to solve a single resource, leads to a nonlinear system of fixed-point equations for the approximate blocking probabilities of each customer, which can be solved iteratively; see S. P. Chung and K. Ross "Reduced Load Approximations for Multi-Rate Loss Networks", IEEE Transactions on Communications volume 41, pages 1222-1231, 1993; F. P. Kelly, "Loss Networks", Annals of Applied Probability, volume 1, pages 319-378, 1991; and W. Whitt, "Blocking When Service is Required from Several Facilities Simultaneously", AT&T Technical Journal, volume 64, pages 1807-1856, 1985.

This reduced-load fixed-point approximation scheme can be applied to large resource-sharing models with the UL and GM bounds by using the blocking probability computer to calculate the exact blocking probabilities for each customer on each resource. Moreover, it is sometimes possible to improve the quality of these reduced-load fixed-point approximations by exploiting the process of the present invention for exactly solving models with more than one resource. We can then decompose the initial large resource-sharing model into smaller submodels (now typically with more than one resource) which can be efficiently solved exactly. We can also act as if these submodels are independent, and reduce the offered load of each customer at each model by the blocking experienced by that customer in other submodels. (When alternative routing is present, it is not possible to exactly solve submodels with more than one resource using the blocking probability computer. Hence submodels are intended for other applications.)

When more than one resource is used in the submodels, it is necessary to determine which resources should be

grouped together in the submodels. For this purpose, we have found that it is advantageous to identify good decompositions for these new reduced-load approximations. We estimate how tightly coupled each pair of resources is by computing the correlations between the resource occupancies in the associated unlimited-capacity model. Unlimited-capacity models have a long tradition in teletraffic theory, e.g., associated with the concept of peakedness and normal approximations; see W. Whitt "Heavy-Traffic Approximations for Service Systems with Blocking", AT&T Bell Laboratories Technical Journal, volume 63, pages 689-708, 1984 and Z. Dziong and J. W. Roberts, (cited above). We thus try to put two resources in the same submodel when they are relatively tightly coupled, i.e., are high correlated.

It is assumed that very lightly loaded resources which provide essentially no constraint are already eliminated from the model. Suppose that p resources remain. Partition the p resources into s subsets, indexing the subsets by k. (Each resource appears in one and only one subset.) Let resource (k,i) be the i^{th} resource of subset k. Let ρ_k be size of the i^{th} subset. Then the resources are indexed by pairs (k,i) for $1 \le i \le \rho_k$ and $1 \le k \le s$. The first step in the approximation is to regard the s submodels as stochastically independent. They are solved separately. The initial offered traffic to each submodel is the offered traffic in the original model that uses the submodel. A crude initial approximation for the blocking probability of customer j is

$$B_j = 1 - \prod_{k=1}^{s} (1 - B_{kj}) , \qquad (87)$$

where B_{kj} is the blocking probability for customer j at submodel k, assumed to be based directly on the original model data. Equation 87 exploits the independence approximation, because it arises by assuming that the probability of customer j not being blocked overall is equal to the product of the probabilities of it not being blocked at each submodel.

If the submodel blocking probabilities B_{kj} are relatively small, then Equation 87 itself can be an excellent approximation. Otherwise, it may be much better to reduce the arrival rate at each submodel to take account of the blocking experienced at other submodels. Since in general there are state-dependent arrival rates, new reduced state-dependent arrival-rate functions λ^*_{kj} for customer j at submodel k are formed by letting

$$\lambda^*_{kj}(m) = \lambda_j(m) \prod_{\substack{l=1 \\ l \neq k}}^{s} (1 - B_{lj}) . \qquad (88)$$

The reduced arrival-rate function λ^*_{kj} depends on the submodel blocking probabilities B_{lj}, and the submodel blocking probabilities in turn depend on the arrival-rate functions used in the submodels. Hence, it is necessary to find a fixed-point solution to Equations 87 and 88. Initially let $\lambda^*_{kj}(m) = \lambda_j(m)$. Then solve the submodel to obtain the submodel blocking probabilities B_{kj} and use Equation 87 to obtain the first candidate overall blocking probabilities B_j. Then solve Equation 88 to obtain new values of $\lambda^*_{kj}(m)$. Iterate, solving Equations 87 and 88, until there is little change. If necessary, the iteration can be modified to get convergence, e.g., by relaxation methods. For instance, the new value of $\lambda^*_{kj}(m)$ can be a convex combination of the previous value and the candidate new value instead of the candidate new value itself.

The advantage of working with submodels instead of individual resources in the reduced-load approximation is that resources that are tightly coupled can be treated together. It is natural to put resources in the same subset when they are relatively tightly coupled, and to put resources in different subsets when they are relatively weakly coupled.

The key for choosing good decompositions is to obtain a useful estimate of how tightly coupled the resources actually are. A way to do this is to compute the correlations between the steady-state occupancies in an associated infinite-capacity model. The correlation between resources i_1 and i_2 is

$$c(i_1, i_2) = \frac{C(i_1, i_2)}{\sqrt{V(i_1)V(i_2)}} , \qquad (89)$$

where V(i) is the variance and $C(i_1,i_2)$ is the covariance. These in turn can be determined from the customer variances and requirements. Let b_{ij} be the number of resource units required by a customer- j request on resource i, and let v_j be the variance of the number of active requests from customer j. Then

$$V(i) = \sum_{j=1}^{r} v_j b_{ij}^2 \qquad (90)$$

and

$$C(i_1, i_2) = \sum_{j=1}^{r} v_j b_{i_1 j} b_{i_2 j} . \qquad (91)$$

Ways to compute means and variances for individual classes were described above in connection with the normal-approximation algorithm for eliminating lightly loaded resources. Finally, resources that are most highly correlated are put together in the same submodel.

B. Adjustment of Resource Capacity and Traffic Bounds

In accordance with another aspect of the present invention, the previously described blocking probability computer is used to adjust the capacity of a single resource, and the adjustment of associated UL and GM parameters, in response to changes of expected traffic load. (Similar procedures could be used for multiple resources.) As the traffic demands increase or decrease, the existing resource capacity may become inadequate or excessive, and the UL and GM parameters may also no longer be appropriate. Since the resource is limited, it is desirable to use a minimum number of resource units to meet the new demand. For the new traffic load, the goal is to identify such a minimum feasible resource capacity, along with good UL and GM traffic bounds, for satisfying the specified performance requirements. That is, the optimization problem under consideration is to minimize the resource capacity by choosing appropriate UL and GM parameters, while meeting the blocking probability requirements.

The present invention uses a search procedure to systematically explore various candidate parameter settings and identify the best among them. A major component of the search procedure is the use of the blocking probability computer to determine the blocking probabilities for a given set of parameters. Since the number of control variables may be large and the optimization problem does not have established mathematical properties (e.g., monotonicity and convexity), it is difficult to devise an efficient algorithm for finding the optimal solution. Consequently, a heuristic search is used, employing the blocking probability computer to identify a good feasible resource capacity and good UL and GM parameters for the new traffic load. The specific search technique presented below should be viewed as one of many alternative approaches. Other search procedures can be developed by those skilled in the art, using the blocking probability computer as the key element.

The search procedure can be invoked to adjust the resource capacity and the UL and GM parameters as a real-time response to changes of traffic load. The time scale in the real-time response may vary in different applications. A block diagram of a system arranged in accordance with the present invention for making capacity adjustments in a resource sharing system is shown in Fig. 9. A capacity adjustment controller 903 manages the capacity adjustment process. The capacity adjustment process is triggered by inputs either directly from the customers in a customer pool 901 or from a traffic load monitor 902. Customers can indicate their desire to either increase or decrease their grade of service. Traffic load monitor 902 can indicate that the current capacity needs to be either increased or decreased. When a capacity adjustment is deemed appropriate, capacity adjustment controller 903 performs a search procedure, using blocking probability computer 904 to find an appropriate new resource capacity to meet the new requirements. This may require adjustment of the customer UL and GM bounds as well. When the new capacity and bounds are determined, the change is made in the resource 905, and the new parameters are sent to customer database 906, where they are used for the ongoing management of the system.

Consider a resource serving r customers indexed by 1 to r, where each customer-j request demands bj units of the resource. It is assumed that all the b_j's have been appropriately scaled by the greatest common denominator of all the b_j's. With this scaling, the largest common factor of all the b_j's is 1. Each customer j has *nominal* and *conditional* blocking probability requirements, denoted by B_j^* and $\bar{B}_j^*$ respectively. The nominal requirement B_j^* is specified for the normal traffic load; that is, each customer submits requests to the resource at the pre-specified load level (which has been agreed upon between customers and the system). In contrast, the conditional requirement $\bar{B}_j^*$ is specified for some form of overload conditions. It will be assumed here that the offered load of all customers other than customer j is X percent (e.g., 10%) above the pre-specified load level. Other conditional blocking requirements can be treated in a similar way, but some may lead to more computation.

Let the UL and GM parameters for customer j be denoted by U_j and L_j, respectively. These are bounds on the number of requests, not the number of resource units. To guarantee a minimum grade of service in case of extreme overload of all other customers, customer j may have a lower bound L_j^* on its GM parameter, so that at least L_j^* requests of customer j can be served by the system at any time. That is, $L_j \geq L_j^*$. In addition, customer j may have a lower bound U_j^* on its UL parameter, so that $U_j \geq U_j^*$.

Let K be the number of resource units (i.e., the resource capacity). Also let B_j and $\bar{B}_j$ be the nominal and conditional blocking probabilities, respectively. Now, the optimization problem can be formalized as:

Minimize K over K, U_j, and $L_j \in \{0\} \cup Z^+$ for given B_j^*, $\bar{B}_j^*$, U_j^* and L_j^*

such that $B_j \leq B_j^*$, $\bar{B}_j \leq \bar{B}_j^*$, $U_j^* \leq U_j$ and $L_j^* \leq L_j$ for all $j = 1,2,...,r$.

The capacity adjustment process is outlined in Fig. 10. The process starts by constantly monitoring the resource usage and customer demands in step 1001. The system periodically determines when the resource capacity needs adjustment in step 1002. When adjustment is needed, step 1003 determines the available resources and customer requirements. Based on these data, the search procedure attempts to find a good resource capacity by the following

four major steps:

a. Find an upper bound K_u for the resource capacity in step 1004;

b. Find an upper bound U_j for the UL parameter for each customer j in step 1005;

c. Find a lower bound L_j for the GM parameter for each customer j in step 1006; and

d. Apply a local search technique to identify the best local optimal solution in step 1007 based on the "pessimistic solution" found in a) to c) above.

These steps are discussed in turn below. In each step, the blocking probability computer is repeatedly invoked to compute the blocking probabilities for the candidate parameter settings.

(1) An Upper Bound K_u for the Resource Capacity

Based on the nominal offered load, the lower bounds L_j^* for the GM parameters, infinite UL parameters and the resource requirements b_j for each customer j, the normal approximation in Equations 31-35 is used to compute the mean and variance, denoted by M_j and V_j respectively, of the number of resource units occupied by each customer j. Let the average and the variance of the number of resource units occupied by all customers be M and V, which are in turn obtained by $M= \Sigma_{j=1}^{r} M_j$ and $V = \Sigma_{j=1}^{r} V_j$, respectively.

The upper-bound capacity K_n for the nominal traffic load can be determined by the following steps:

(a) Set a search parameter γ to 4.

(b) Set $K=M+\gamma\sqrt{V}$. Using the nominal offered load, the resource capacity K and the GM parameter L_j^* and an infinite UL parameter for each customer j, invoke the blocking probability computer to compute the blocking probability B_j for each customer j.

(c) If there exists at least one j from 1 to r where $B_j > B_j^*$, increase γ by 1 and continue with step b). Otherwise, set $K^* = M+\gamma\sqrt{V}$ and continue with step d).

(d) Reset γ to 1.

(e) Set $K = M +\gamma\sqrt{V}$. Using the nominal offered load, the resource capacity K and the GM parameter L_j^* and an infinite UL parameter for each customer j, invoke the blocking probability computer to compute the blocking probability B_j for each customer j.

(f) If there exists at least one j from 1 to r where $B_j \leq B_j^*$, decrease γ by 1 and continue with step e). Otherwise, set $K* = M+\gamma\sqrt{V}$ and continue with step g).

(g) By repeatedly invoking the blocking probability computer, conduct a bisection search to look for K_u between K* and K* such that $B_j \leq B_j^*$ for all j from 1 to r and all $K^* \geq K \geq K_u$, but there exists at least one j from 1 to r that $B_j > B_j^*$ for $K=K_u-1$.

The computation of resource occupancy M and V, and the above steps (a) to (g) are repeated for the conditional (overload) traffic load, to find an upper-bound capacity K_c for the overload condition. For simplicity, this can be done by assuming that *all* customers have X% overload. Then, it is not necessary to do a separate calculation for each customer. Finally, the upper-bound resource capacity K_u is chosen to be the maximum of K_n and K_c.

(2) An Upper Bound for the UL Parameter of Each Customer

The idea here is to find an upper bound U_j for the UL parameter U_j for each customer j based on the assumption that the resource has the upper-bound capacity K_u obtained above. These upper bounds are determined first by considering the nominal offered load and the nominal blocking probability requirements. Then, the same procedure is repeated for the overload and conditional blocking probability requirements. The final upper bound U_j for customer j is set to be the maximum of the bounds for the nominal and overload condition, and the lower bound U_j^* pre-specified by the customers.

The upper bounds for the UL parameters at the nominal traffic condition can be determined by the following steps:

(a) Based on the nominal offered load, the upper bound K_u for the capacity, the lower bounds L_j^* for the GM parameter, UL parameters equal to K_u and the resource requirements b_j for all customers j, invoke the blocking probability computer to determine the blocking probabilities for all customers. These computed blocking probabilities are referred to as the *referenced blocking probabilities* in the following description.

(b) Assume that the GM and UL parameters for each customer j are L_j^* and K_u, respectively. By (29), compute the average m_j and the variance v_j of the number of requests in service from each customer j.

(c) Define a set C to be {1,2,...r}. Set γ to an upper-bound value (e.g., 6). This upper-bound value can be verified by using the blocking probability computer to make sure that the blocking probability for customer j is less than or equal to Y% (e.g., 10%) of the blocking probability requirement plus the referenced blocking probability for all customers j from 1 to r.

(d) For each $j \in C$, set $U_j = m_j + \gamma\sqrt{v_j}$.

(e) Using the resource capacity K_u, and the GM and UL parameters L_j^* and U_j for each customer j, invoke the blocking probability computer to obtain the blocking probabilities for all customers.

(f) If the blocking probability for customer j is less than or equal to Y% of the blocking probability requirement plus the referenced blocking probability for all customers j from 1 to r, decrease γ by 1 and continue with step d). If not, proceed with step g).

(g) Identify customer j where the computed blocking probability based on γ is greater than Y% of the blocking probability requirement plus the referenced blocking probability for the customer, but it is less for $\gamma+1$. Let S be the set of all such customers. (Clearly, $S \subseteq C$.)

(h) Select one $j \in S$, and set $U_{j^*} = m_j + \gamma\sqrt{v_j}$ and $U_j^* = m_j + (\gamma+1)\sqrt{v_j}$. In addition, for each $k \in S$ with $k \neq j$, set $U_k = m_k + \gamma\sqrt{v_k}$.

(i) Perform a bisection search for U_j between U_{j^*} and U_j^* such that, for any UL parameter of customer j being less than U_j, the blocking probability for customer j is greater than Y% of the blocking probability requirement plus the referenced blocking probability for the customer. At each step of the bisection search, the blocking probability computer is invoked to determine the blocking probability for customer-j requests with all other UL parameters, U_k's for $k \neq j$, the GM parameters L_j^*'s and the offered load unchanged.

(j) Repeat steps h) and i) for each of all other $j \in S$.

(k) Set C = C - S. If C is non-empty, decrease γ by 1 and continue with step d). Otherwise, the upper bounds $\{U_j; j=1,2,...,r\}$ for the corresponding UL parameters are thus obtained.

As pointed out earlier, this procedure of steps a) to k) is repeated for the overload traffic and conditional blocking probability requirements. To reduce computation, this can be done by assuming *all* customers have X% overload. Then, the final upper bound U_j for the UL parameter for each customer j is obtained by taking the maximum of the respective bounds for the nominal and overload conditions, and the lower bound U_j^* pre-specified by the customers.

(3) A Lower Bound for the GM Parameter of Each Customer

The basic idea for identifying a lower bound for the GM parameter for each customer is similar to that presented above for the upper bounds for the UL parameters. Since the upper bounds for the UL parameters U_j's have been determined, they can be used in determining the lower bounds for the GM parameters here. For sake of completeness, the steps for the GM bounds for the nominal traffic load are given as follows:

(a) Based on the resource capacity K_u, the nominal offered load, the resource requirement b_j, the lower bound for the GM parameter L_j^*, the upper bound for the UL parameter U_j for all customers j, invoke the blocking probability computer to determine the blocking probabilities for all customers. These blocking probabilities are referred to as the *referenced blocking probabilities* in the following.

(b) Assume that the GM and UL parameters for each customer j are L_j^* and U_j, respectively. By (29), compute the average m_j and the variance v_j of the number of requests in service from each customer j.

(c) Define a set C to be {1,2,...,r}. Set β to a lower-bound value (e.g., -6). This lower-bound value can be verified by using the blocking probability computer to make sure that the blocking probability for customer j is less than or equal to Y% (e.g., 1%) of the blocking probability requirement plus the referenced blocking probability for all customers j from 1 to r.

(d) For each $j \in C$, set $L_j = \max \{ 0, m_j + \beta\sqrt{v_j} \}$.

(e) Using the resource capacity K_u, and the GM and UL parameters L_j and U_j for each customer j, invoke the blocking probability computer to obtain the blocking probabilities for all customers.

(f) If the blocking probability for customer j is less than or equal to Y% of the blocking probability requirement plus the referenced blocking probability for all customers j from 1 to r, increase β by 1 and continue with step d). Otherwise, proceed with step g) below.

(g) Identify customer j where the computed blocking probability for β is greater than Y% of the blocking probability requirement plus the referenced blocking probability for the customer, but it is less for β-1. Let S be the set of all such customers. (Clearly, $S \subseteq C$.)

(h) Select one $j \in S$, and set $L_{j^*} = \max\{ 0, m_j + (\beta-1)\sqrt{v_j} \}$ and $L_j^* = \max \{ 0, m_j + \beta\sqrt{v_j} \}$. In addition, for each $k \in S$ with $k \neq j$, set $L_k = \max\{0, m_k + \beta\sqrt{v_k} \}$.

(i) Perform a bisection search for L_j between L_{j^*} and L_j^* such that, for any GM parameter nor customer j being greater than L_j, the blocking probability for customer j is greater than Y% of the blocking probability requirement plus the referenced blocking probability for the customer. At each step of the bisection search, the blocking probability computer is invoked to determine the blocking probability for customer-j requests with all other GM parameters, L_k's for $k \neq j$, all UL parameters U_j's and the offered load unchanged.

(j) Repeat steps h) and i) for each of all other $j \in S$.

(k) Set $C = C - S$. If C is non-empty, increase β by 1 and continue with step d). Otherwise, the lower bounds $\{L_j; j=1,2,...r\}$ for all GM parameters are thus obtained.

Similar to the upper bounds for the UL parameters, this procedure of steps a) to k) is repeated for the overload traffic and conditional blocking probability requirements. For simple computation, this can be done by assuming *all* customers have X% overload. Then, the final lower bound L_j for the GM parameter for each customer j is obtained by taking the maximum of the respective bounds for the nominal and overload conditions, and the lower bound L_j^* prespecified by the customers.

(4) Search for the Best Local Optimal Solution

The solution obtained so far with the resource capacity K_u, the UL bounds $\{U_j\}$ and the GM bounds $\{L_j\}$ is a "pessimistic" solution. It is feasible, because all nominal and conditional blocking probability requirements are satisfied, but it remains to see if it can be improved. The purpose of this local search procedure is to use the feasible solution as a starting point to look for a local optimum.

At a high level, the main idea of the search procedure is to attempt to reduce the resource capacity from the upper bound K_u by checking to see if there exists a feasible UL and GM parameter setting, which satisfies the nominal and conditional blocking probability requirements. If there exists such a parameter setting, a better solution is found and the procedure repeats itself to further reduce the resource capacity. Otherwise, the procedure stops, as it cannot further reduce the capacity, and the "best" feasible solution has been obtained.

Let K, U_j and L_j be the resource capacity, and the UL and GM parameters for each customer j of the candidate solution under consideration by the local search algorithm. The search procedure is given as follows:

(a) Record the best solution obtained so far: $K = K_u$, $U_j = U_j$ and $L_j = L_j$ for each j=1 to r.

(b) Empty a list of 2r-tuples and then put the 2r-tuple of UL and GM parameters $(U_1,..., U_r,L_1,...,L_r)$ as the first entry onto the list. Set K=K-1.

(c) Consider a candidate solution: the resource capacity K, and the UL and GM parameters U_j and L_j for each customer j.

(d) For the new candidate capacity and traffic parameters, invoke the blocking probability computer to obtain the nominal and conditional blocking probabilities (denoted by B_j and $\bar{B}_j$ respectively) for each customer j at the nominal and overload traffic conditions.

(e) If all blocking probability requirements are met, i.e., if $B_j \leq B_j^*$ and $\bar{B}_j \leq \bar{B}_j^*$ for all j=1 to r, then the tested capacity K is feasible. In this case, record the tested solution as the new best solution and continue with step b). Otherwise, continue with step f).

(f) Set η_j to be the maximum of B_j/B_j^* and $\bar{B}_j/\bar{B}_j^*$ for each customer j. Set η^* and η_* to be the minimum and maximum of all η_j for j=1 to r, respectively. If $\eta_* \geq 1/\eta^*$, continue with step g). Otherwise, proceed with step h).

(g) The fact that $\eta_* \geq 1/\eta^*$ implies that a customer has an exceptionally high blocking probability (either nominal or conditional). Let j be the customer index that yields the maximum η_j among all customers. If $U_j < U_j$, increase U_j by 1. If $U_j = U_j$, increase L_j by 1. Continue with step i).

(h) The fact that $\eta_* < 1/\eta^*$ implies that a customer has an exceptionally small blocking probability (either nominal or conditional). Let j be the customer index that yields the minimum η_j among all customers. If $L_j > L_j$, decrease L_j by 1. If $L_j = L_j$, decrease U_j by 1.

(i) Check if the 2r-tuple of the UL and GM parameters $(U_1,..., U_r,L_1,...,L_r)$ is already included on the list. If not, add the tuple to the list as a new entry and continue with step c). Otherwise, the procedure cannot further reduce the capacity and the local optimal solution has been found, as previously recorded by step (a) or (e).

C. Response to Resource Failures

A block diagram of a system arranged in accordance with the present invention for performing traffic diversion in response to a resource failure in a resource sharing system is shown in Fig. 11. There are two main components: (1) a set of resources 1100 and (2) a Centralized Operations Center (COC) designated as 1110. A resource status monitor 1101 detects whenever one or more of the resources fails. In a telecommunications system with links as resources, the link status is typically monitored in the switches within the network. A link failure can be detected therein by, for example, a loss of signal.

When a resource failure is detected, resource status monitor 1101 sends a signal to a traffic diversion controller 1105 within COC 1110, which controls the process of diverting demand from the failed resource to alternative resources. Since resource failure is regarded as a temporary situation, it may be deemed desirable to base the traffic diversion on recent customer load on the resources, rather than (or in addition to) negotiated parameters. Hence, it is necessary to know the recent load on the system. For this purpose, a traffic load measurement system 1102 records customer usage over time. Such a system would normally be available anyway, e.g., for the purpose of billing and resource management (including capacity adjustment as described above in Section B).

Periodically, traffic load measurement system 1102 sends traffic load data to a traffic load database 1104 within COC 1110. Thus, at the time of resource failure, traffic diversion controller 1105 has access to the recent traffic loads from traffic load database 1104. When a failure occurs, traffic diversion controller 1105 determines possible sets of alternative resources (e.g., alternative routes) that could be used to meet the demand that was being met by the failed resource. For this purpose, traffic diversion controller 1105 obtains necessary data about availability of alternative resources from a routing database 1106. Then, traffic diversion controller 1105 invokes blocking probability computer 1107 to determine how much load can be diverted to each set of alternative set of resources. Blocking probability computer 1107 is needed to determine if the blocking probabilities of all customers, original and diverted, after the diversion will be satisfactory. Finding the appropriate amount to divert requires extra analysis, which is described below. Traffic diversion controller 1105 also determines appropriate upper limit (UL) and guaranteed minimum (GM) bounds to protect both the original traffic and the diverted traffic on these alternative resources. When traffic controller 1105 determines an appropriate diversion process, this diversion process is communicated to diversion traffic bound enforcement system 1103 for enforcement. For example, in a telecommunication network, diversion bound enforcement could be performed by the switches in the network. Traffic bound enforcement system 1103 thus ensures that the

controls are implemented.

In accordance with yet another aspect of the present invention, blocking probability computer 1107 is used to divert customer requests from a failed resource, so that at least part of the demands can be met by the remaining resources. In a resource-sharing system, each customer request occupies certain units of different resources for a period of time. When a resource fails, customer requests demanding the failed resource will be blocked. However, it is assumed that the remaining resources can accommodate some of the needs of these customers.

Such use of alternative resources to meet customer demand is common. In the example of a circuited-switch telecommunication network in Fig. 1, when a link (i.e., resource) fails for some reason such as a fiber cut, calls can be routed through other links, bypassing the failed link. In many cases, the total number of resource units needed to satisfy a specific customer request in case of resource failure is higher than in normal conditions; e.g., because the alternative route often has more links. Nevertheless, the use of alternative resources provides great flexibility in efficiently sharing the resources among customers when failures occur.

The problem addressed by this aspect of the invention is: Given a resource failure, how can the system effectively divert to the remaining resources future customer requests that originally demand the failed resource, without adversely affecting the quality of service provided to other customers? (It is assumed that those customer requests receiving service from the failed resource at the time of the failure are lost, but some of these requests may re-appear as part of future demands.)

An obvious solution to the problem is to divert the customer requests from the failed resource to other applicable resources that have sufficient spare capacities. However, the way to do this is not clear. First, the available spare capacities are actually not known in advance, because the traffic and the performance requirements are characterized probabilistically. In addition, a resource failure may occur together with an increase of the arrival rates of customer requests for the resource. For example, events such as earthquakes, flooding, and hurricanes damaging a telephone link can also cause a large number of calls to be made to the disaster area, thus putting more demand for the failed link. Such a potentially large increase of customer requests for the failed resource, which are now diverted to the remaining resources, can seriously degrade the quality of service for the original customers. Moreover, there is no guaranteed minimum grade of service for the diverted customers.

The central idea of this aspect of the invention is to divert an appropriate *proportion* of the expected traffic load of customer requests from the failed resource to alternative resources. Using a fairness criterion, an attempt is made to divert the same proportion of traffic for all diverted customers. However, alternative schemes could be considered, which might give different priorities to different customers. The methods here would also apply to these other schemes with appropriate modification.

In addition to diverting an appropriate proportion of the load, the procedure adjusts the UL and GM parameters to try to guarantee the same grade of service for the diverted traffic of each customer. Finally, two new sets of UL bounds are established to ensure the overall protection of the original and diverted traffic against overload of each other.

The diversion process starts by considering one set of alternative resources that can serve the diverted traffic. The procedure determines how much can be diverted to this first set of resources. If there is still more traffic to be diverted, then the procedure considers another set of alternative resources that can serve the diverted traffic. The diversion process continues until all applicable alternative resources have been considered or the traffic load of the affected customers has all been diverted to alternative resources. As with other aspects of this invention, the key component of this diversion procedure is the blocking probability computer. It is used to determine the amount of traffic to divert to alternative resources, and the associated UL and GM traffic parameters.

To illustrate the steps of this diversion procedure in response to failures, a generic resource-sharing system and a telecommunication network are considered separately in the following description. The main purpose of the generic system is to explain the general concepts of the traffic diversion in a relatively simple setting, so that the main ideas are easy to grasp. The purpose of the telecommunication network example is to demonstrate the feasibility for one important class of applications.

Consider a generic resource-sharing system with two resources, one containing squares and the other containing circles, indexed by 1 and 2. Let there be four customers. In normal situations, customer 1 and 2 requests each need one square, and customer 3 and 4 requests each need one circle. Suppose that appropriate UL and GM bounds have been assigned to each customer. Let these be denoted by U_j and L_j for customer j. Unlike in the rest of the detailed description, throughout this Section C, the *UL and GM bounds are specified in terms of resource units* instead of requests. Thus, U_j is b_j times what it is in other sections.

It is assumed that when the square resource fails, the circle resource can serve as an alternative, but each request from customer 1 and 2 requires two circles. In case of such failure, it is desirable to identify and enforce appropriate UL and GM traffic bounds for the circles such that requests of customer 1 and 2, the *diverted traffic,* can be accommodated as much as possible, while protecting the satisfactory grade of service for customer 3 and 4 requests, the *original traffic.* One can view the diversion of customer 1 and 2 requests to the circle resource as the assignment of two new customers to the circle resource in response to the failure of the square resource.

In this generic resource-sharing system, the traffic diversion procedure constantly monitors the offered load of requests of each customer. Let the estimated offered load of customer j requests immediately prior to the failure of the square resource be ρ_j. (This estimated offered load may reflect both the original load contracted by the customer and the actual observed load over recent history prior to the failure.) When the square resource fails, the respective resource requirements for customer 1 to 4 requests for the circle resource are $b_{21}=2$, $b_{22}=2$, $b_{23}=1$ and $b_{24}=1$. Then, the procedure attempts to divert a fraction, $0 \le \alpha \le 1$, of offered load of customer 1 and 2 requests to the circle resource.

Since the new candidate offered load is different from the original offered load for the diverted traffic, it is natural to adjust the UL and GM bounds. A simple approach would be to make these bounds proportional to the old bounds, in proportion to the new offered load $\alpha\rho_j$ compared to the contracted offered load. However, to better meet the blocking requirements, the normal approximation can be used. The normal approximation accounts for the way the capacity should change when the offered load changes, so that the blocking probabilities remain fixed (approximately). Hence, the new UL and GM bounds for this diverted traffic (denoted by U_j and L_j for j = 1 and 2) for the circle resource can be determined by the normal approximation. That is, for j = 1 and 2, let the number of circles required by the diverted customer-j requests be approximated by a normally distributed random variable, $N(m_j, \sigma_j^2)$, with mean m_j equal $\alpha\rho_j b_{2,j}$ and variance σ_j^2 being $\alpha\rho_j z_j b_{2,j}^2$ where z_j is the arrival peakedness of customer-j requests. Let c and d be two real-valued parameters, properly chosen according to the blocking probability requirements and other engineering considerations. (Typical values of c and d lie between 2 and 4, and -4 and -2, respectively.) If ρ_j is the original contracted offered load on which the UL and GM bounds U_j and L_j were originally assigned, then c and d can be chosen so that $U_j = m_j + c\sigma_j$ and $L_j = m_j + d\sigma_j$ for $\alpha = 1$. Then, for customer j=1 and 2, the new UL parameter U_j' is chosen to be the minimum of the positive integer closest to $m_j + c\sigma_j$ and $(K_2 - L_3 - L_4)$, where K_2 is the total number of circles available. Furthermore, the new GM parameter L_j' is set to be the minimum of the integer closest to $m_j + d\sigma_j$ and $(K_2 - L_3 - L_4)$.

Combining the original and diverted traffic, the new candidate offered load for the circle resource is given by a vector, $\underline{\rho} = (\alpha\rho_1, \alpha\rho_2, \rho_3, \rho_4)$ and the associated UL and GM bounds have been obtained above. Now the procedure invokes the blocking probability computer to determine the blocking probability for requests of each customer in sharing the circle resource. These probabilities are compared to the blocking probability requirements. Depending on whether or not all the requirements are satisfied, the procedure can be repeated to attempt to divert more or less customer 1 and 2 requests to the circle resource by a simple binary search on α. (The candidate UL and GM bounds are adjusted for each new α.) Eventually, the procedure stops after diverting an appropriate fraction of customer 1 and 2 requests to the circle resource with the associated UL and GM parameters specified. It may be desirable to further adjust and refine the UL and GM bounds for the diverted traffic, using the blocking probability computer.

In addition to the individual UL bound for the diverted traffic from each customer, the procedure offers further overload protection by obtaining two new UL bounds (denoted by U_o and U_d) for the original traffic (customer 3 and 4 requests) and the diverted traffic (customer 1 and 2 requests). Since different customers with possibly very different resource requirements are combined for these new UL bounds, it is important that these bounds be expressed in terms of resource units rather than requests. These new UL bounds U_o and U_d can be found by normal approximations as follows. Let the number of circles occupied by the original traffic be approximated by a normally distributed random variable, $N(m_o, \sigma_o^2)$, with mean m_o equal $\Sigma_{j=3}^{4} \rho_j b_j$ and variance σ_o^2 being $\Sigma_{j=3}^{4} \rho_j z_j b_j^2$. Then, U_o is chosen to be the minimum of the positive integer closest to $m_o + c\sigma_o$ and $(K_2 - L_3 - L_4)$ where c is an appropriate parameter, typically in the range of 2 to 4, and not necessarily equal to the parameter c above. Similarly, U_d for the diverted customer 1 and 2 requests can be obtained. The new parameter c might again be different from previous ones. For example, it may be deemed desirable to give greater protection to the original traffic than to the diverted traffic. These UL bounds U_o and U_d can be checked and refined by using the blocking probability computer.

Until the square resource is put back in service, the system diverts each new customer 1 and 2 request to the circle resource for service, while the UL and GM traffic bounds for each customer, as well as the new UL bounds for the overall original and diverted traffic are continuously enforced (as described above). This way, not only customer 1 and 2 requests are served in case of failure of the needed square resource, the grades of service for all customers are also guaranteed and protected by the UL and GM bounds.

The basic principle of protecting both the diverted and original traffic in face of resource failures can work in many systems in essentially the same way as in the generic example above. However, the identification of alternative resources for the failed resources may be more involved than in this generic example. Furthermore, the enforcement of the newly established UL and GM bounds is also likely to be more complicated. To show the applicability of the proposed approach to a specific system, a telecommunication network is considered. This network can be a current circuit-switched network or a future B-ISDN network based on the ATM technology. A high-level flowchart for traffic diversion in response to link failures in the network is outlined in Figure 12 and the diversion procedure is discussed in detail below.

Consider a telecommunication network such as shown in Fig. 1, with a number of communication links, which are the resources in question. To ensure a high degree of reliability, there often are multiple routes (i.e., a path consisting of multiple links) from one switch (node) to another in the network. Thus, each call can possibly be routed through one of several routes and the choice is made by a "routing algorithm" at the time of the call setup. Some routing algorithms

commonly used in telecommunication networks, such as the Real-Time Network Routing algorithm described by Ash et al. in U.S. Patent 5,101,451, issued March 31, 1992, are called "dynamic" algorithms, because they make use of the current network state to select the most appropriate route for each call. The specific routing algorithm is not critical here; it could be static or dynamic.

To illustrate the applicability of this invention, routing algorithms are classified as *non-adaptive* and *adaptive* in their response to failures as follows. In non-adaptive routing, when one link of a route from one switch to another fails, the entire route must be replaced. Calls for the origin-destination switch pair are routed via other preestablished alternative routes. In contrast, an adaptive routing algorithm automatically looks for a new route between the two end switches (referred to as *the failure-end switches)* of the failed link. As a result, calls can be routed according to the original route until reaching a failure-end switch. Then, the calls are directed to the new alternative route, bypassing the failed link to the other failure-end switch. Finally, the call routing proceeds from there along the remaining part of the original route, until reaching their destination. It is possible that there are multiple new routes connecting the failure-end switches. In this case, each call in question can take one of these routes according to the routing algorithm in use, if applicable. Alternatively, a round-robin scheme, a sequential overflow scheme (where the new routes are arranged in a priority order, such that low priority routes are attempted for routing the call when high priority routes are busy), or other state-dependent dynamic scheme can be used for this purpose.

If the network under consideration employs a non-adaptive routing algorithm, it will be efficient for the source switch to divert its calls from the failed route to other alternative routes and to enforce the UL and GM traffic bounds for performance guarantee and protection. In this case, the traffic diversion should be carried out on the basis of origin-destination switch pairs. On the other hand, for adaptive routing algorithms, it may be more efficient for the failure-end switches to divert traffic from the failed link to alternative routes connecting the switches and to enforce the associated UL and GM bounds for the diverted calls. Then, traffic is diverted simply on a per-call-class basis, without considering the source and destination of calls. The following discussion focuses on the traffic diversion and enforcement only for the adaptive routing algorithms. The same principle also applies to the non-adaptive routing algorithms, but with added complexity in traffic monitoring, diversion, and enforcement on an origin-destination basis.

Assume that the telecommunication network has a *centralized operations center (COC)* for traffic monitoring, diversion and other maintenance functions, block 1110 in Fig. 11. Also assume that critical information for traffic diversion is available at the COC. This information includes: a) the expected call load for each origin-destination switch pair in the network, b) the preferred routes for calls for each switch pair, and c) the grade of service in terms of blocking probabilities, the contracted (offered) load, and the UL and GM bounds for each call class on each link. With the contracted load in the network, these UL and GM parameters are specified and enforced to guarantee a minimum grade of service and to protect against overload of each class of calls.

Assuming that a link i carries r classes of calls, let the contracted load for the link be denoted by $\hat{\underline{\rho}}_o = \hat{\rho}_o^1, \hat{\rho}_o^2, \ldots, \hat{\rho}_o^r)$. In addition, let the UL and GM bounds in terms of the number of circuits for the contracted traffic for link i be $\underline{U}_o \equiv (U_o^1, U_o^2, \ldots, U_o^r)$ and $\underline{L}_o \equiv (L_o^1, L_o^2, \ldots, L_o^r)$, respectively. Based on $\hat{\underline{\rho}}_o$, $\underline{U}_o$ and $\underline{L}_o$ for each link i, the COC pre-computes the *binding parameters* c_j and d_j for each call class j as

$$c_j = \frac{U_o^j - \hat{\rho}_o^j b_{ij}}{\sqrt{\hat{\rho}_o^j z_j b_{ij}^2}} \text{ and } d_j = \frac{L_o^j - \hat{\rho}_o^j b_{ij}}{\sqrt{\hat{\rho}_o^j z_j b_{ij}^2}},$$

where z_j is the arrival peakedness of class j calls and b_{ij} is the number of circuits occupied by a class j call on link i.

The traffic diversion process starts with step 1201 of the flowchart of Fig. 12, in which each switch in the network constantly monitors the amount of traffic load of different call classes (i.e., the *carried load)* on each link emerging from the switch and the fraction of calls blocked (i.e., the *blocking probabilities).* Each switch reports the carried load and blocking probabilities for each of its links to the COC periodically and the COC saves the data for future referencing.

When a switch detects a link failure in step 1202 by loss of signal from the link, it reports the failure to the COC. Let switch A and B be the failure-end switches for the failed link. After receiving the notification of the failure, the COC retrieves in step 1203, the most current data of the carried load and blocking probabilities for the failed link. It is assumed that the COC can develop a good estimate of the offered load for each traffic class, based on the original contracted offered load and recent history of carried load. The estimated load could simply be the original contracted load (for which no data are needed) or the most recent estimate of carried load, but it could be a more complicated combinations of these and other historical data. For simplicity, here it is assumed that only the most recent carried load estimate is used.

Let there be r classes of calls sent from switch A to B via the failed link and let the call classes be indexed by 1,2,...,r. Furthermore, let the carried load and the blocking probability for each call class j for the failed link be denoted by ϕ_j and P_j, respectively, for j=1,2,...,r. Following the data retrieval, the COC estimates the amount of offered load of class i calls on the failed link from switch A to B, ρ_f^j, by dividing ϕ_j by 1-P_j also in step 1203. Note that this offered load is expected to be different from the contracted load for the link because of traffic fluctuation. The COC may thus elect

to adjust ρ_f^j, e.g., replace ρ_f^j by the minimum or some other function of the observed (carried) load and the contracted load for that class. Let $\underline{\rho}_f \equiv (\rho_f^1, \rho_f^2, ..., \rho_f^r)$ be the amount of traffic to be *diverted.*

Next, in step 1204, a determination is made as to whether $\underline{\rho}_f$, the amount of traffic remaining to be diverted, has reached zero. If YES, the diversion procedure has successfully been completed, and the process stops in step 1207. Otherwise, the COC proceeds to step 1205, in order to identify the next shortest alternative route from switch A to switch B, which has not been used already in this traffic diversion. It may be deemed desirable to identify the shortest alternative route in terms of the number of intermediate switches the route has to go through. Then one can apply a technique such as that invented by Mansour and Nguyen in the U.S. Patent Number 5,058,105, to find the shortest alternative route. If it is determined in step 1206 that no new alternative route exists, then all alternative routes from switch A to B, bypassing the failed link, have been considered and loaded with a portion of the diverted traffic. In this case, the process of Fig. 12 stops in step 1207, although it cannot fully divert all traffic from the failed link without degrading the grade of service for the original traffic in the network.

If a new alternative route exists, so that a YES result occurs in step 1206, the COC invokes the blocking probability computer in step 1208 to determine the amount of traffic load of classes 1 to r, denoted by $\underline{\rho}_d = (\rho_d^1, \rho_d^2, ..., \rho_d^r)$, that the alternative route can accept from the failed link. In particular, this method uses a proportional approach to divert traffic of all classes; that is, the same proportion of calls from all classes is diverted from the failed link to an alternative route. Furthermore, as an approximation step in the analysis, it is assumed here that the original traffic for all classes on all links of the contemplated route are mutually independent Thus, in the model, only the diverted calls require the simultaneous possession of all links of the alternative route. (Another approach is to consider the whole network with all call classes, instead of focusing on the alternative route. However, in order to analyze a large network, the blocking probability computer is likely to apply reduced-load approximations, which also ignore some degree of the resource dependence.)

The proportion of traffic to be diverted to an alternative route is determined by the following steps:

(a) Let the chosen alternative route consist of J links, indexed by 1,2,...,J. The COC estimates the offered load for each call class on each of these links by dividing the carried load by one minus the blocking probability for the call class on the link. (The carried load and blocking proability on each link are updated and recorded at the COC periodically.) As with diverted traffic, this may be the minimum or some other function of the contracted and measured offered load for the link. For simplicity, each link is assumed to have the same number r of call classes. Let the offered load for one of the J links be denoted by $\underline{\rho}_o = (\rho_o^1, \rho_o^2, ..., \rho_o^r)$.

(b) For the chosen alternative route, construct a resource-sharing model with J resources and (J+1)r classes of calls (customers), where each resource represents one of the J links in the alternative route, and the j^{th} and $(J+1)^{st}$ set of r classes of requests correspond to the original and diverted traffic on the j^{th} link of the route, respectively. As stated above, it is assumed that the original traffic classes on all links are mutually independent and only the diverted calls require the simultaneous possession of all J links. Let α_U and α_L denote the respective upper and lower bounds for the proportion of traffic to be diverted from the failed link. Initially, set $\alpha_L=0$ and $\alpha_U=1$.

(c) Set $\alpha=(\alpha_U-\alpha_L)/2$ and the amount of diverted traffic to the alternative route, $\underline{\rho}_d$, to be $\alpha\underline{\rho}_f = (\alpha\rho_f^1, \alpha\rho_f^2, ..., \alpha\rho_f^r)$.

(d) If the difference between α_U and α_L is less than a pre-specified tolerance, α is the fraction of traffic to be diverted to the alternative route. That is, $\underline{\rho}_d$ is finalized to be $(\alpha\rho_f^1, \alpha\rho_f^2, ..., \alpha\rho_f^r)$. In addition, the per call-class UL and GM parameters U_d^j and L_d^j for call class j=1 to r, and the new UL bounds $U_{o,i}$ and $U_{d,i}$ for the original and diverted traffic on each link i have been found. The procedure stops. Otherwise, proceed with step e) below.

(e) For each link i of the alternative route, set γ_i to be the ratio of expected circuit occupancy of the diverted traffic to that of the contracted traffic on link i. That is,

$$\text{set } \gamma_i = \frac{\sum_{j=1}^{r} \rho_d^j b_{ij}}{\sum_{j=1}^{r} \hat{\rho}_o^j b_{ij}}.$$

Let the UL and GM bounds for the diverted traffic on link i be $\underline{U}_d \equiv (U_d^1, U_d^2, ..., U_d^r)$ and $\underline{L}_d \equiv (L_d^1, L_d^2, ..., L_d^r)$. Let c_j and d_j be the pre-determined per-call-class binding parameters for the UL and GM bounds for link i described above. For each call class j, set U_d^j to be the minimum of the positive integer closest to

$$(\gamma_i \hat{\rho}_o^j b_{ij} + c_j \sqrt{\gamma_i \hat{\rho}_o^j z_j b_{ij}^2})$$

and K_i, where K_i is the total number of circuits in link i. In addition, set L_d^j to be the minimum of the positive integer closest to

$$(\gamma_i \hat{\rho}_o^j b_{ij} + d_j \sqrt{\gamma_i \hat{\rho}_o^j z_j b_{ij}^2})$$

and

$$\left\lfloor \frac{(K_i - \sum_{k=1}^{r} L_o^k) L_o^j}{\sum_{k=1}^{r} L_o^k} \right\rfloor .$$

(f) For each call class j, choose the minimum among all U_d^j's for all links i of the alternative route. Without introducing additional notation, let U_d^j denote such minimum. Similarly, for each class j, obtain the minimum L_d^j among the L_d^j's for all links i. (The UL and GM traffic bounds U_d^j and L_d^j can be enforced for class j calls on every link of the alternative route.)

(g) In order to provide further protection against overload, obtain two sets of new UL bounds (denoted by $U_{o,i}$ and $U_{d,i}$ for every link i) for the original traffic and the diverted traffic by normal approximations as follows. For each link i of the alternative route, let the number of circuits occupied by the original (contracted) traffic on link i be approximated by a normally distributed random variable, $N(m_o,\sigma_o^2)$, with mean m_o equal $\sum_{j=1}^{r} \hat{\rho}_o^j b_{ij}$ and variance σ_o^2 being $\sum_{j=1}^{r} \hat{\rho}_o^j z_j b_{ij}^2$. With properly chosen binding parameters c and d, $U_{o,i}$ is chosen to be the minimum of the integer closest to $m_o+c\sigma_o$ and $K_i - \sum_{j=1}^{r} (L_o^j + L_d^j)$. Similarly, $U_{d,i}$ is set to be the minimum of the integer closest to $m_o + d\sigma_o$ and $K_i - \sum_{j=1}^{r} (L_o^j + L_d^j)$. The bounds $U_{o,i}$ and $U_{d,i}$ can be checked and tuned by using the blocking probability computer.

(h) With the combined offered load, $\underline{\rho}_o$ and $\underline{\rho}_d$, and the per-call-class UL and GM parameters, and the overall UL bounds for the original and diverted traffic, invoke the blocking probability computer to obtain the blocking probability for each call class, original and diverted, in the resource-sharing modeL Compare the blocking probability for each call class with the pre-specified blocking requirement (which is known to the COC). In this comparison, the same blocking requirements can be applied to both the original and diverted calls, provided that they are of the same class. If there is any call class with blocking probability exceeding the requirement, set $\alpha_U=\alpha$. (As an option, α_U is set to α only after the procedure has attempted to adjust the UL and GM bounds for the diverted traffic with other parameters unchanged, and found the blocking probability for at least call class violating the requirement. This option may improve the diversion efficiency for some parameter settings because the UL and GM bounds estimated by the normal approximation may not be appropriate for the amount of diverted traffic under consideration.) Otherwise, set $\alpha_L = \alpha$. Then, continue the binary search on α with step c).

After identifying the amount of traffic to be diverted to the chosen alternative route in step 1208 and the UL and GM parameters, the COC notifies the failure-end switch A and other switches along the alternative route with the associated traffic bounds. Then, the COC computes the remaining traffic to be diverted from the failed link in step 1209, and the process returns to step 1204 to divert the remaining traffic, if any. This procedure continues until all traffic is successfully diverted from the failed link or all alternative routes have been examined and loaded with certain amount of the diverted traffic.

The per-call-class UL and GM bounds and the new UL bounds for the original calls can be enforced by the switches on the alternative route. More precisely, the bounds associated with a link can be observed by the switch that sends traffic onto the link. In contrast, the per-call-class UL and GM bounds and the new UL bound for the diverted calls can be enforced only by the failure-end switch A. Thus, switch A only need to know the corresponding minimum of the UL and GM bounds for various call classes or links for proper traffic enforcement for the diverted calls.

One way to enable efficient traffic enforcement is to keep the link failure known to only the failure-end switches and the COC. As a result, other switches in the network can continue to route traffic as if the link failure did not occur. When a call needs to be routed from switch A to B (where switch A can be the originating or via switch of the call), switch A can identify if the call is a diverted call (i.e., one requires the failed link). If so, switch A makes sure that the UL and GM traffic bounds are satisfied according to the steps for traffic enforcement, as described above, before accepting and diverting the call to one of the established alternative routes, according to the call routing algorithm used in the network. if applicable. Alternatively, a round-robin scheme, a sequential overflow scheme, or a state-dependent dynamic scheme, can be used to make the selection. Note that the routing scheme can affect the peakedness of the call arrivals to the alternative routes. Thus, for a given the routing scheme, it may be desirable to adjust the peakedness parameters for various alternative routes for input to the blocking probability computer for accurate determination of the blocking probabilities.

If the call routing algorithm of the network requires the use of the loading status of the failed link, and if it is not available for the diverted traffic directly, then such information can be assembled by the failure-end switches, by keeping track of the loading status of diverted traffic among the established alternative routes.

With this traffic diversion method, one can view that a failed link is partially or even fully "replaced" by the established alternative routes, because the GM parameters guarantee a minimum grade of service for the diverted traffic on the alternative routes, and the new UL parameters protect the original and diverted traffic along the alternative route from possible overload of each other.

In fact, this traffic diversion procedure can handle multiple link failures and switch failures. Note that a switch failure

can be regarded as a case of multiple (simultaneous) link failures for all links emerging from the switch. For the case of multiple link failures that do not occur simultaneously, the diversion procedure can work well, because traffic can be diverted from one failed link at a time on a first-come-first-served basis. If a particular link is involved in several alternative routes for different failed links, the link can be loaded with traffic diverted from multiple failed links. As a result, the link carries an additional set of call classes diverted from each failed link. In this situation, the diverted traffic from various failed links will have their respective UL and GM traffic bounds, which can be enforced separately by the corresponding switches, in the same way as described above.

To divert traffic in response to a switch failure, the procedure requires each switch to monitor not only the traffic load to its neighboring switches (i.e., traffic load on each of the links emerging from the switch) as suggested above, but also the amount of traffic routed to the neighbors of the neighboring switches. The traffic load is periodically reported to the COC. As a result, when a switch fails, the amount of traffic between each pair of its neighboring switches is known to the COC. Then, the same diversion procedure can be applied to divert this amount of traffic from one of the neighboring switches to another, bypassing the failed switch. Now, the failure-end switches are the neighboring switches. In addition, the alternative routes to receive the diverted traffic are those from one switch to another that used to be the neighbors of the failed switch. The procedure determines the amount of diverted traffic and the associated UL and GM bounds for each of the alternative routes. As before, the UL and GM traffic bounds can be enforced by the switches along the alternative routes.

Furthermore, a minor variation of the same procedure also applies to a telecommunication network without a COC. In this case, each switch monitors and keeps the traffic load data locally. When a link failure occurs, a distributed algorithm, such as the distributed asynchronous Bellman-Ford algorithm (see Bertsekas and Gallager, *Data Networks,* Prentice-Hall, 1992, pp.404-410), can be used to identify the shortest alternative routes. Once the alternative route is found, the failure-end switch A is responsible for collecting the traffic data needed for the diversion from the switches on the alternative route. With the traffic data, switch A performs the traffic diversion in the same way as the COC does. As a result, the amount of diverted traffic and the associated UL and GM bounds can be determined for the alternative route. Then, switch A notifies the switches on the route with the bounds for traffic enforcement. If switch A finds that there is traffic remaining to be diverted from the failed link, it initiates further traffic diversion on the next alternative route. The traffic enforcement by the UL and GM bounds for the original traffic can continue to be done by the switches that send traffic onto the links. In addition, the traffic bounds for the diverted traffic can be enforced by the failure-end switch A.

Various modifications and adaptations of the present invention will be apparent to persons skilled in the art. Accordingly, it is intended that this invention be limited only by the following claims.

Claims

1. A method of controlling admission of new customers to a shared resource that is serving existing customers, said method comprising the steps of

   determining the blocking probability requirements of said existing customers;
   determining the blocking probability requirements and grade of service desired by each of said new customers;
   determining if said new customers' requirements can be satisfied without violating said requirements of said existing customers, said determination being made in accordance with a product-form resource sharing model which has a normalization constant; and
   if said new customers requirements can be satisfied, allowing said new customer to be admitted to said shared resource, wherein said last mentioned determining step includes
   calculating the blocking probabilities for usage of said shared resource by said new customers and said existing customers by numerically inverting the generating function of said normalization constant of said resource sharing model.

2. A method of controlling admission of a new customer to a shared resource serving existing customers, each of said customers having a predetermined traffic load, said method comprising the steps of

   determining nominal and conditional blocking probability requirements for each of said existing customers and for said new customer; said nominal blocking probability requirements being based upon compliance by all of said customers with said predetermined traffic load and said conditional blocking probability requirements being based upon deviation by one or more of said customers from said predetermined traffic load,
   determining if both said new customer's nominal and conditional blocking probability requirements can be satisfied without violating both said nominal and conditional blocking probability requirements of said existing

customers,
if both said new customers nominal and conditional blocking probability requirements can be satisfied, allowing said new customer to be admitted to said shared resource.

3. A method of controlling admission of new customers to a shared resource, in which said new customers can be provided with multiple grades of service, including protection against overloads from existing customers and other new customers, said method comprising the steps of

retrieving stored information defining the requirements of said existing customers, and obtaining the requirements and grade of service desired by each of said new customers;
responsive to said retrieved information, determining if said new customer's requirements can be satisfied without violating the requirements of said existing customers, said determination being made in accordance with a product-form resource sharing model which has a normalization constant; and
allowing said new customer to be admitted to said shared resource with said desired grade of service only if said new customers requirements can be satisfied
wherein said last mentioned determining step includes
calculating the blocking probabilities for usage of said shared resource by said new customers and said existing customers by numerically inverting the generating function of said normalization constant of said resource sharing model.

4. A method of controlling admission of a new customer to a shared resource serving existing customers, in which said existing customers and said new customer can be provided with multiple grades of service, each of said grades of service being defined by a traffic requirement and a blocking requirement, said method comprising the steps of

obtaining information indicating (a) the grade of service of said existing customers and (b) the grade of service desired by said new customer;
responsive to said grade of service desired by said new customer, assigning UL and GM bounds on the usage of said shared resource to said new customer,
determining if said new customer's grade of service can be provided without violating the requirements of said existing customers, and
if said new customer's grade of service can be provided, allowing said new customer to be admitted to said shared resource with said desired grade of service,
wherein said determining step includes computing the blocking probability for said new customer, and determining if said blocking probability satisfies the UL and GM bounds assigned to said new customer.

5. A method for real time admission control to determine in real time whether a prospective new customer can be admitted to a resource with a desired grade of service, comprising the steps of

determining the desired grade of service for the prospective customer,
determining if the desired grade of service can be met considering both the prospective customer and all existing customers,
if the desired grade of service can be provided, admitting the new customer with the desired grade of service, and
if the desired grade of service cannot be met, then determining if a lower grade of service is feasible.

6. A method of adjusting the capacity of a shared resource so as to satisfy new levels of customer demand, comprising the steps of

responsive to a change of customer demand, determining the new demand level for said resource, and adjusting the capacity to satisfy the blocking requirements for all customers,
wherein said adjusting step includes calculating the blocking probabilities for usage of said shared resource by said customers by numerically inverting the generating function of said normalization constant of said resource sharing model.

7. A method of diverting customer usage from failed resources to alternative resources in a shared resource system, comprising the steps of

monitoring the system to determine failure of a resource,
identifying appropriate alternative resources, and
adjusting usage to meet blocking probability requirements of existing and diverted customers on said alternative resources,
wherein said adjusting step is made in accordance with a product-form resource sharing model which has a normalization constant, and
wherein the blocking probabilities for usage of said shared resource by said diverted customers and said existing customers are calculated by numerically inverting the generating function of said normalization constant of said resource sharing model.

8. A technique for providing different grades of service and protection against overloads to customers sharing a resource, comprising the steps of

assigning each customer upper-limit (UL) and guaranteed-minimum (GM) bounds on its requests, said upper limit bound putting an upper limit on the number of requests from that customer that can be in service at any time, and said guaranteed-minimum bound guaranteeing that there will always be available resource units in the resources to serve a specified number of requests from that customer, thereby creating a resource-sharing model having a product-form steady-state distribution, and
solving the resource-sharing model using a blocking probability computer (BPC) which operates by

(a) directly expressing said model in terms of normalization constants (or partition function values) appearing in the product-form steady-state distribution,
(b) computing the normalization constants by constructing the generating function of the normalizing constant and
(c) numerically inverting the generating function.

9. The method of claim 8 wherein said numerically inverting step includes

using a Fourier-series method with a scaling algorithm tailored to the resource-sharing model.

10. The invention defined in claim 8 or 9 wherein said BPC is arranged to

(a) approximately determine the traffic load on each resource using a normal approximation scheme,
(b) determining if there are some resources that are so lightly loaded that they provide essentially no constraint,
(c) responsive to said last mentioned step, eliminating lightly loaded resources from said model before constructing said generating function.

11. The invention defined in claim 8 or 9 further including reducing the effective dimension of the generating function by conditional decomposition and inverting the variables of the multidimensional generating function in an appropriate order.

12. The method of claim 11 further including determining, after said normal approximation and conditional decomposition steps, if the model is not yet solvable, and if so, performing a reduced-load fixed-point approximation process to approximately solve the model.

FIG. 1

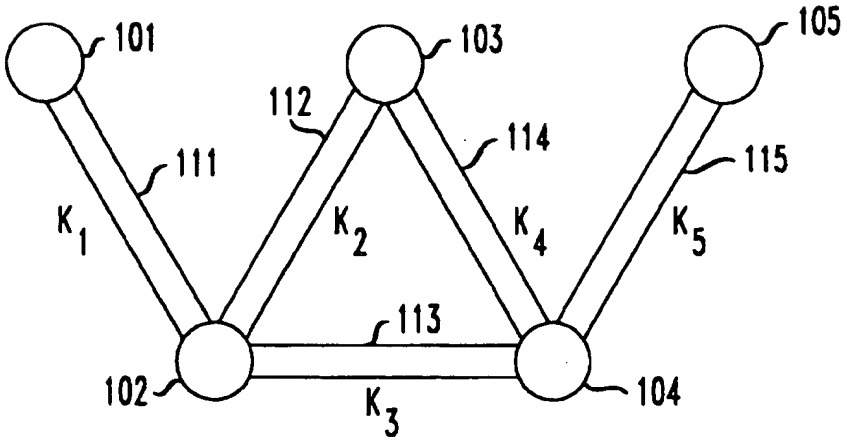

101
103
105
112
114
111
115
K_1
K_2
K_4
K_5
113
102
104
K_3

FIG. 2

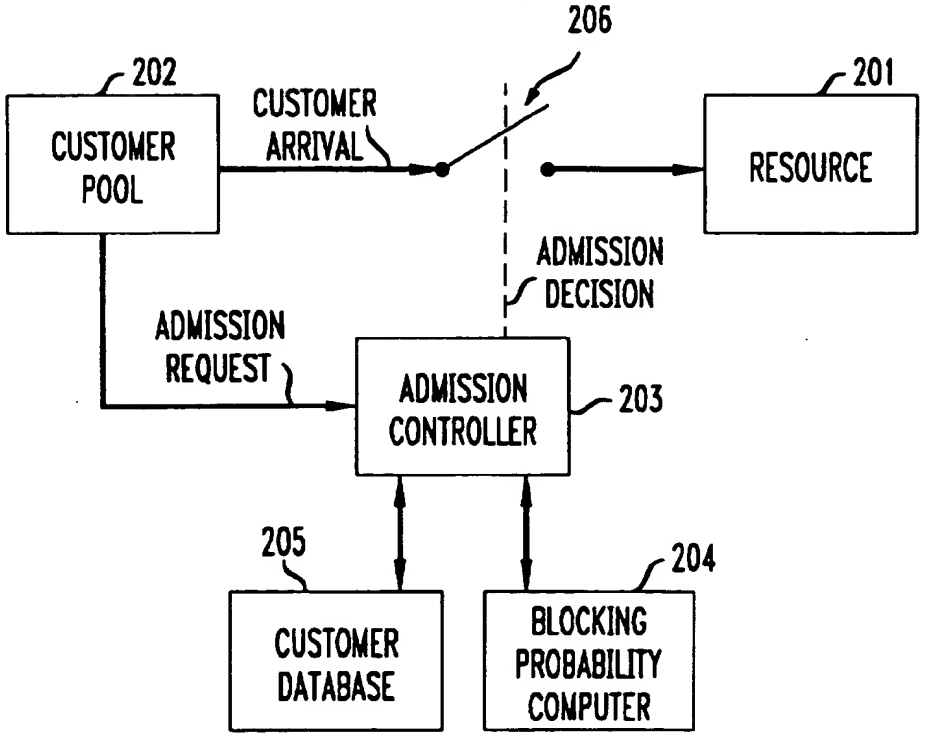

206
202
201
CUSTOMER POOL
CUSTOMER ARRIVAL
RESOURCE
ADMISSION DECISION
ADMISSION REQUEST
ADMISSION CONTROLLER
203
205
204
CUSTOMER DATABASE
BLOCKING PROBABILITY COMPUTER

FIG.3

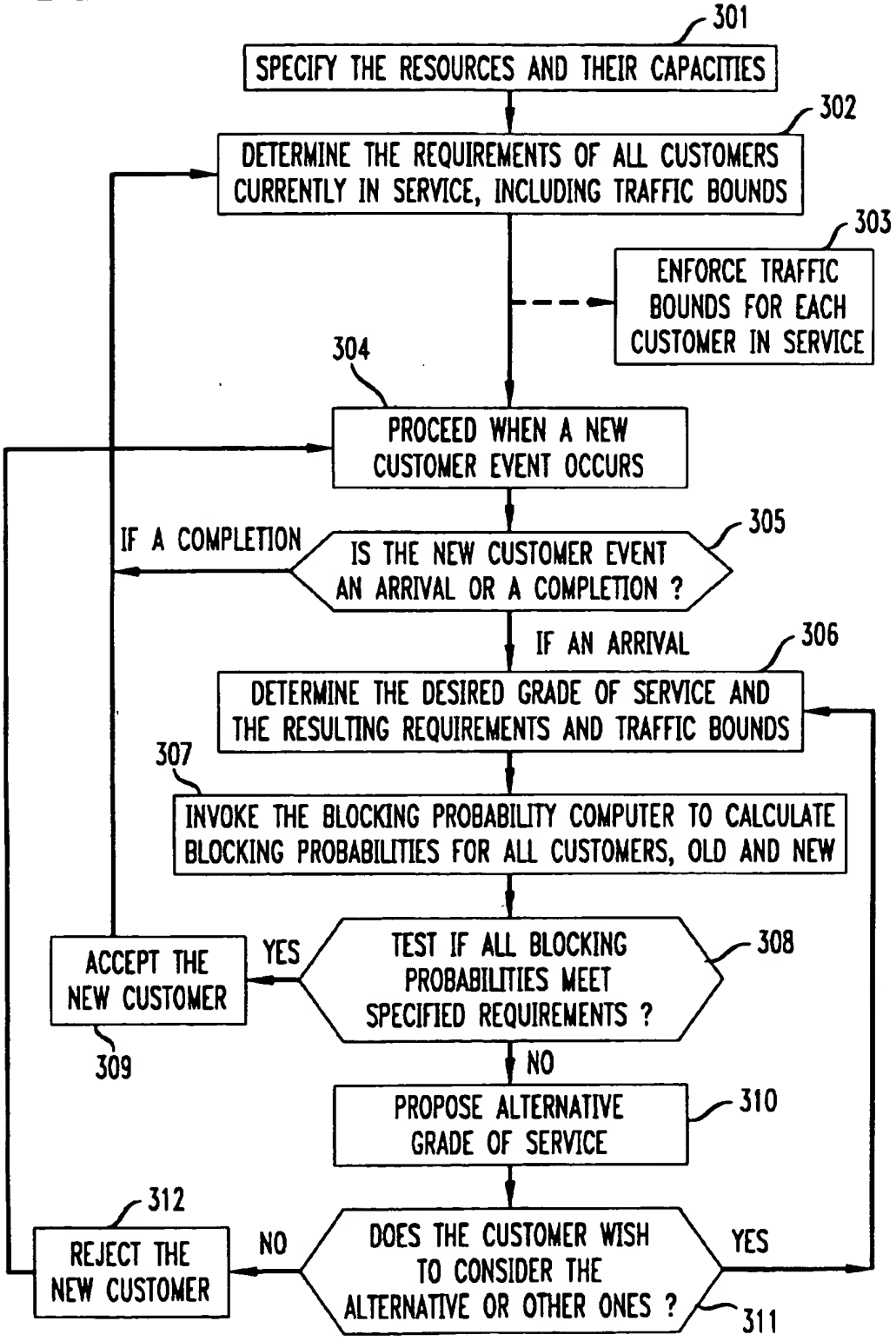

301
SPECIFY THE RESOURCES AND THEIR CAPACITIES
302
DETERMINE THE REQUIREMENTS OF ALL CUSTOMERS CURRENTLY IN SERVICE, INCLUDING TRAFFIC BOUNDS
303
ENFORCE TRAFFIC BOUNDS FOR EACH CUSTOMER IN SERVICE
304
PROCEED WHEN A NEW CUSTOMER EVENT OCCURS
IF A COMPLETION
305
IS THE NEW CUSTOMER EVENT AN ARRIVAL OR A COMPLETION ?
IF AN ARRIVAL
306
DETERMINE THE DESIRED GRADE OF SERVICE AND THE RESULTING REQUIREMENTS AND TRAFFIC BOUNDS
307
INVOKE THE BLOCKING PROBABILITY COMPUTER TO CALCULATE BLOCKING PROBABILITIES FOR ALL CUSTOMERS, OLD AND NEW
YES
308
TEST IF ALL BLOCKING PROBABILITIES MEET SPECIFIED REQUIREMENTS ?
ACCEPT THE NEW CUSTOMER
309
NO
310
PROPOSE ALTERNATIVE GRADE OF SERVICE
312
REJECT THE NEW CUSTOMER
NO
DOES THE CUSTOMER WISH TO CONSIDER THE ALTERNATIVE OR OTHER ONES ?
YES
311

FIG.4

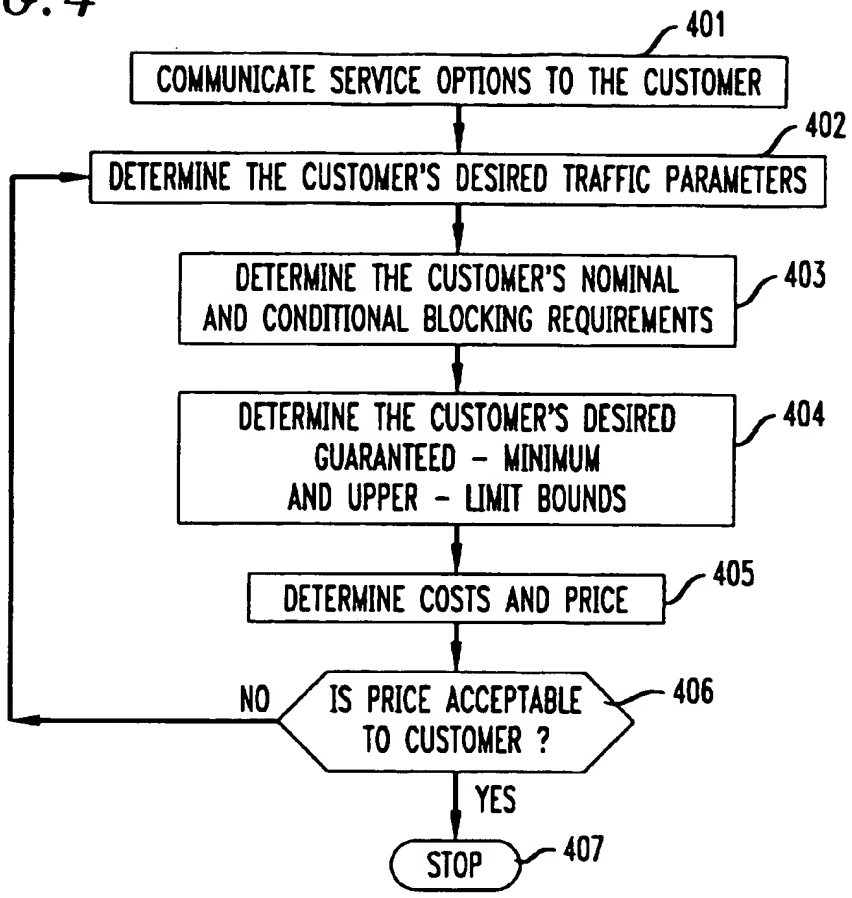

401
COMMUNICATE SERVICE OPTIONS TO THE CUSTOMER
402
DETERMINE THE CUSTOMER'S DESIRED TRAFFIC PARAMETERS
DETERMINE THE CUSTOMER'S NOMINAL AND CONDITIONAL BLOCKING REQUIREMENTS
403
DETERMINE THE CUSTOMER'S DESIRED GUARANTEED - MINIMUM AND UPPER - LIMIT BOUNDS
404
DETERMINE COSTS AND PRICE
405
NO
IS PRICE ACCEPTABLE TO CUSTOMER ?
406
YES
STOP
407

FIG.5

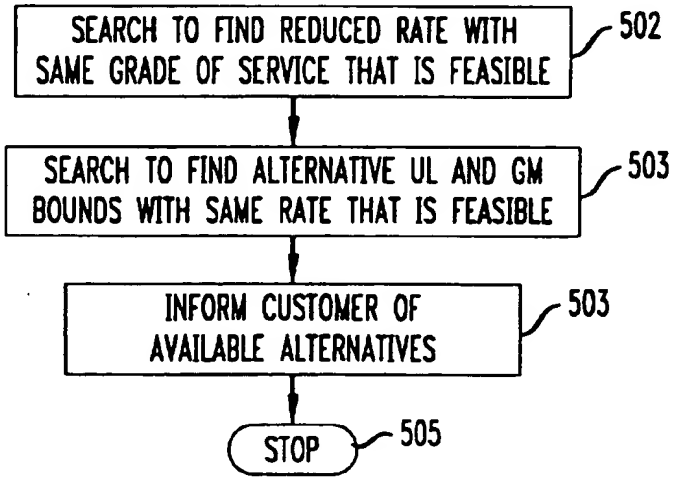

SEARCH TO FIND REDUCED RATE WITH SAME GRADE OF SERVICE THAT IS FEASIBLE
502
SEARCH TO FIND ALTERNATIVE UL AND GM BOUNDS WITH SAME RATE THAT IS FEASIBLE
503
INFORM CUSTOMER OF AVAILABLE ALTERNATIVES
503
STOP
505

FIG. 6

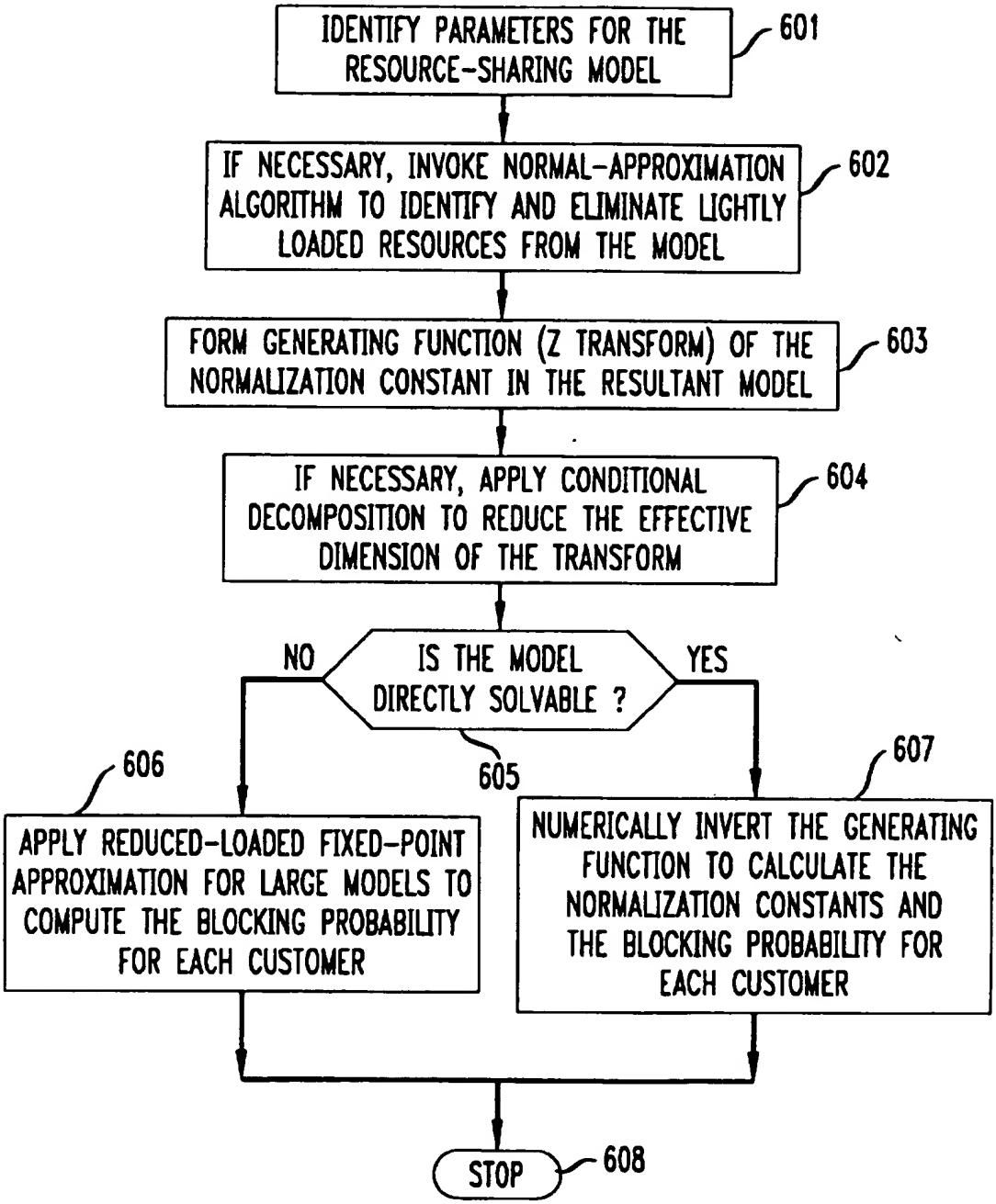

IDENTIFY PARAMETERS FOR THE RESOURCE-SHARING MODEL
601
IF NECESSARY, INVOKE NORMAL-APPROXIMATION ALGORITHM TO IDENTIFY AND ELIMINATE LIGHTLY LOADED RESOURCES FROM THE MODEL
602
FORM GENERATING FUNCTION (Z TRANSFORM) OF THE NORMALIZATION CONSTANT IN THE RESULTANT MODEL
603
IF NECESSARY, APPLY CONDITIONAL DECOMPOSITION TO REDUCE THE EFFECTIVE DIMENSION OF THE TRANSFORM
604
IS THE MODEL DIRECTLY SOLVABLE ?
NO
YES
605
606
APPLY REDUCED-LOADED FIXED-POINT APPROXIMATION FOR LARGE MODELS TO COMPUTE THE BLOCKING PROBABILITY FOR EACH CUSTOMER
607
NUMERICALLY INVERT THE GENERATING FUNCTION TO CALCULATE THE NORMALIZATION CONSTANTS AND THE BLOCKING PROBABILITY FOR EACH CUSTOMER
STOP
608

FIG. 7

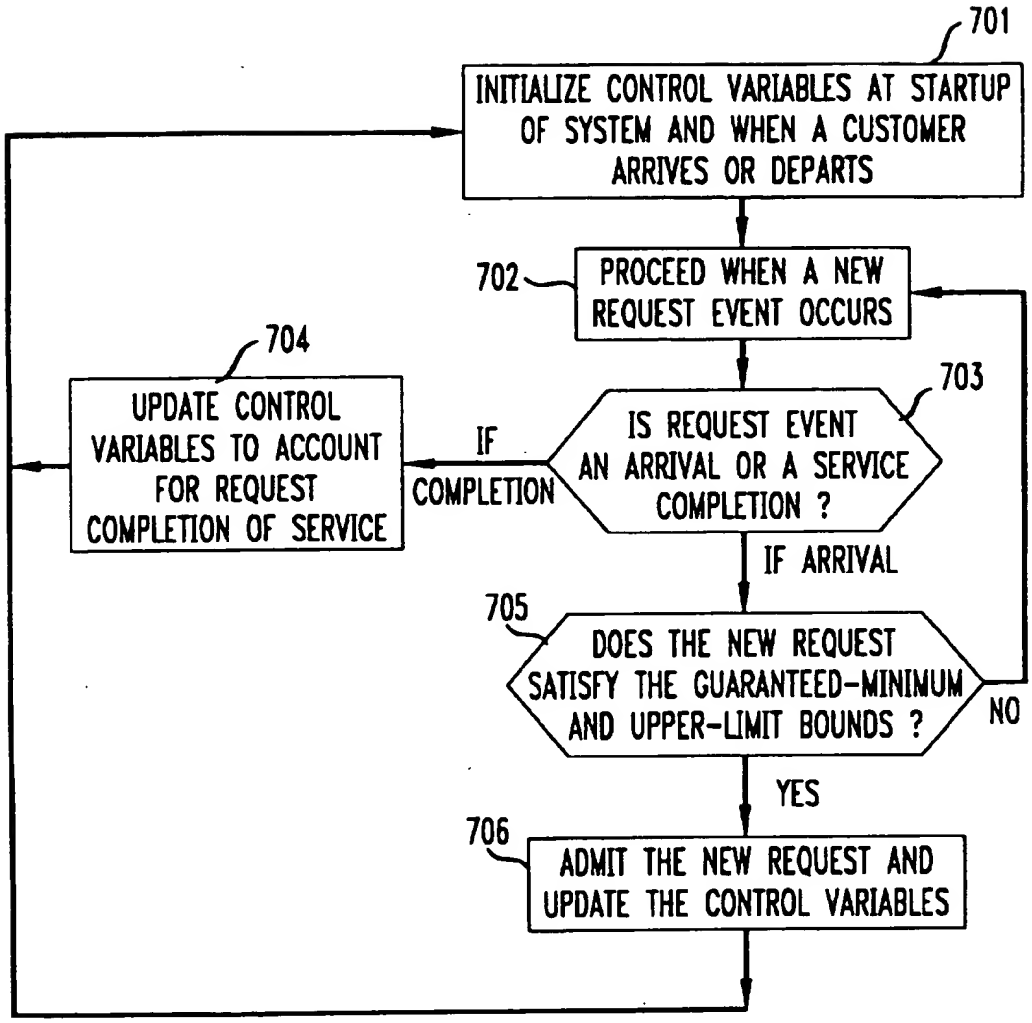

701
INITIALIZE CONTROL VARIABLES AT STARTUP OF SYSTEM AND WHEN A CUSTOMER ARRIVES OR DEPARTS
702
PROCEED WHEN A NEW REQUEST EVENT OCCURS
703
IS REQUEST EVENT AN ARRIVAL OR A SERVICE COMPLETION ?
IF COMPLETION
704
UPDATE CONTROL VARIABLES TO ACCOUNT FOR REQUEST COMPLETION OF SERVICE
IF ARRIVAL
705
DOES THE NEW REQUEST SATISFY THE GUARANTEED-MINIMUM AND UPPER-LIMIT BOUNDS ?
NO
YES
706
ADMIT THE NEW REQUEST AND UPDATE THE CONTROL VARIABLES

FIG.8

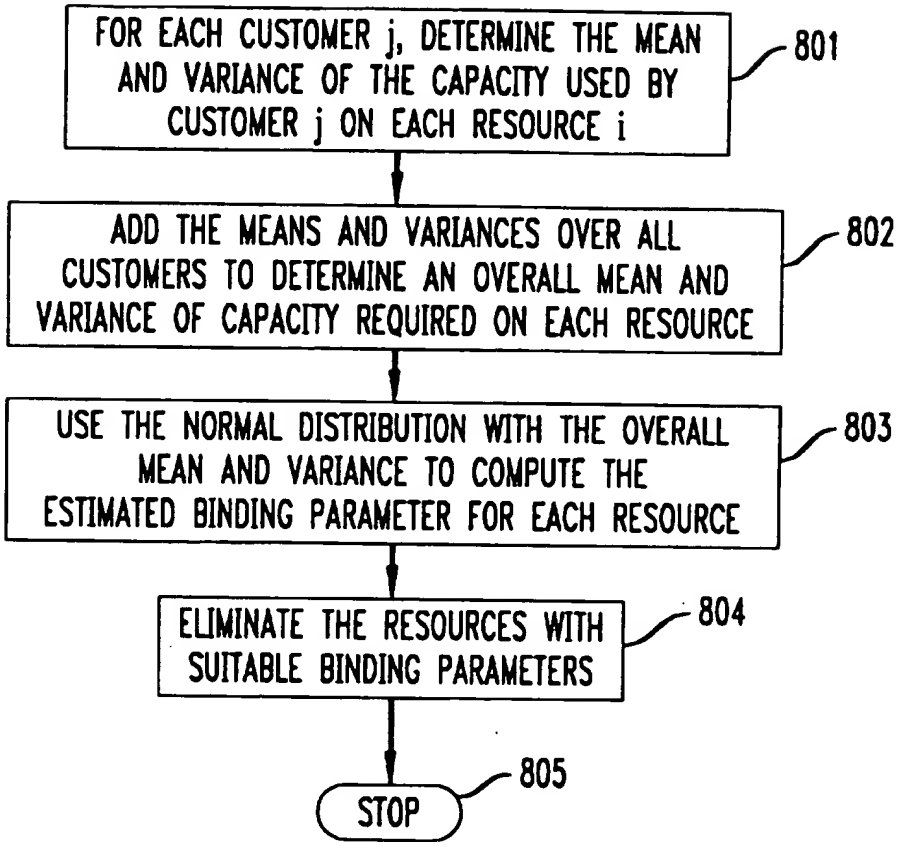

FOR EACH CUSTOMER j, DETERMINE THE MEAN AND VARIANCE OF THE CAPACITY USED BY CUSTOMER j ON EACH RESOURCE i
801
ADD THE MEANS AND VARIANCES OVER ALL CUSTOMERS TO DETERMINE AN OVERALL MEAN AND VARIANCE OF CAPACITY REQUIRED ON EACH RESOURCE
802
USE THE NORMAL DISTRIBUTION WITH THE OVERALL MEAN AND VARIANCE TO COMPUTE THE ESTIMATED BINDING PARAMETER FOR EACH RESOURCE
803
ELIMINATE THE RESOURCES WITH SUITABLE BINDING PARAMETERS
804
STOP
805

FIG. 9

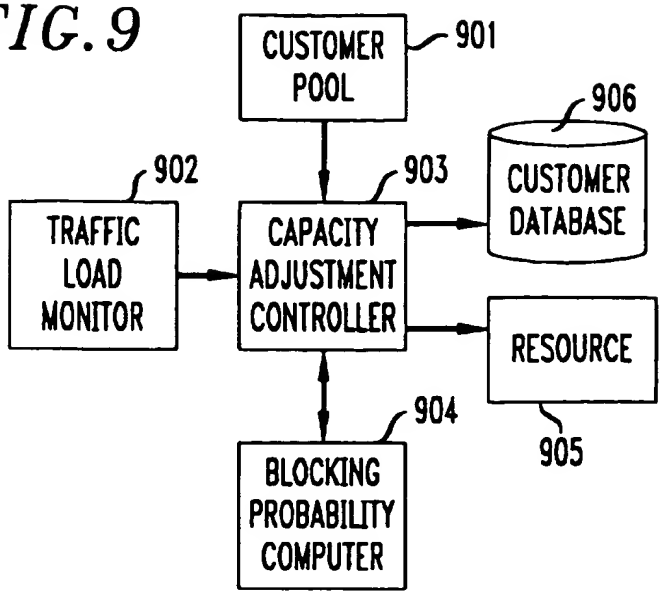

CUSTOMER POOL
901
906
CUSTOMER DATABASE
902
TRAFFIC LOAD MONITOR
903
CAPACITY ADJUSTMENT CONTROLLER
RESOURCE
905
904
BLOCKING PROBABILITY COMPUTER

FIG. 10

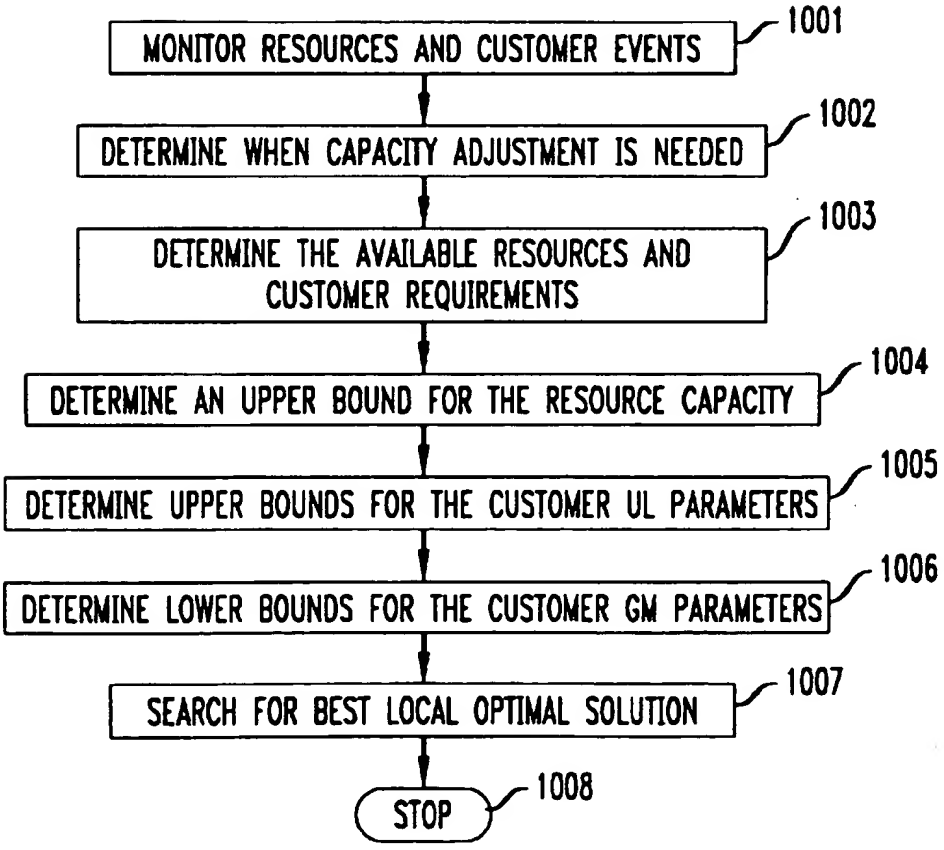

MONITOR RESOURCES AND CUSTOMER EVENTS
1001
DETERMINE WHEN CAPACITY ADJUSTMENT IS NEEDED
1002
DETERMINE THE AVAILABLE RESOURCES AND CUSTOMER REQUIREMENTS
1003
DETERMINE AN UPPER BOUND FOR THE RESOURCE CAPACITY
1004
DETERMINE UPPER BOUNDS FOR THE CUSTOMER UL PARAMETERS
1005
DETERMINE LOWER BOUNDS FOR THE CUSTOMER GM PARAMETERS
1006
SEARCH FOR BEST LOCAL OPTIMAL SOLUTION
1007
STOP
1008

FIG. 11

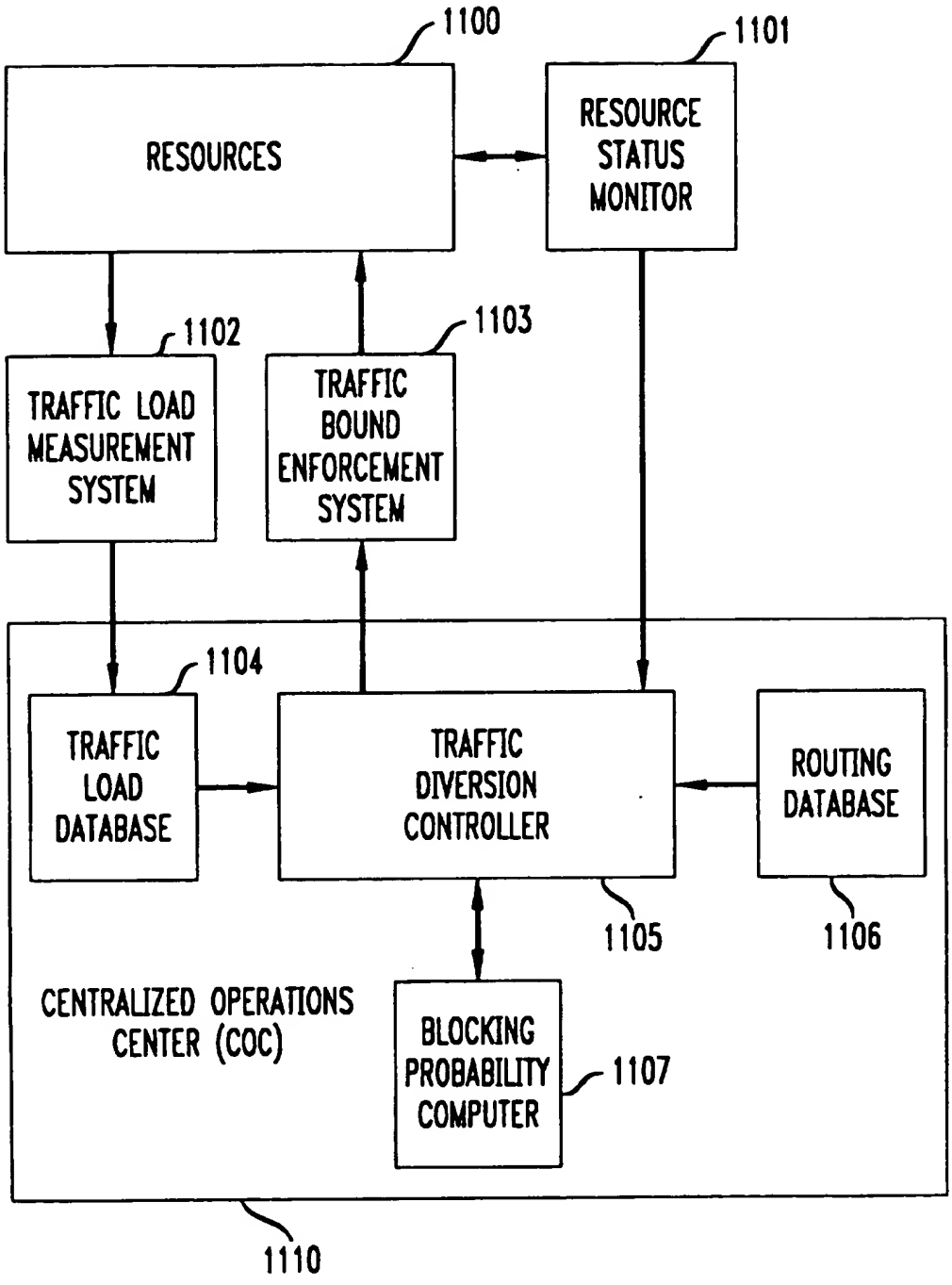
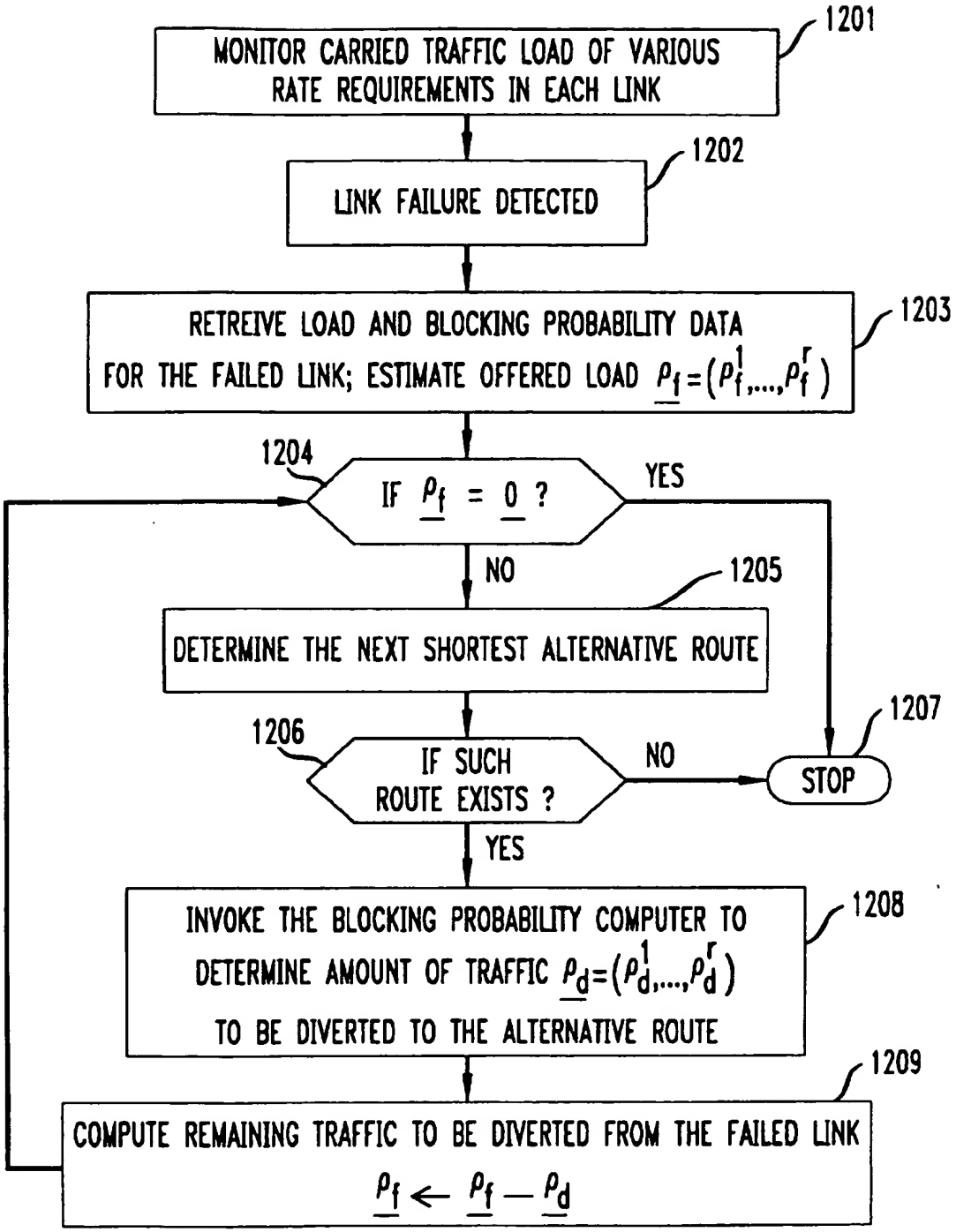

1100
1101
RESOURCES
RESOURCE STATUS MONITOR
1102
1103
TRAFFIC LOAD MEASUREMENT SYSTEM
TRAFFIC BOUND ENFORCEMENT SYSTEM
1104
TRAFFIC LOAD DATABASE
TRAFFIC DIVERSION CONTROLLER
ROUTING DATABASE
1105
1106
CENTRALIZED OPERATIONS CENTER (COC)
BLOCKING PROBABILITY COMPUTER
1107
1110

FIG. 12


1201
MONITOR CARRIED TRAFFIC LOAD OF VARIOUS RATE REQUIREMENTS IN EACH LINK
1202
LINK FAILURE DETECTED
1203
RETREIVE LOAD AND BLOCKING PROBABILITY DATA FOR THE FAILED LINK; ESTIMATE OFFERED LOAD $\underline{\rho_f}=(\rho_f^1,\ldots,\rho_f^r)$
1204
IF $\underline{\rho_f} = \underline{0}$?
YES
NO
1205
DETERMINE THE NEXT SHORTEST ALTERNATIVE ROUTE
1206
IF SUCH ROUTE EXISTS ?
NO
1207
STOP
YES
1208
INVOKE THE BLOCKING PROBABILITY COMPUTER TO DETERMINE AMOUNT OF TRAFFIC $\underline{\rho_d}=(\rho_d^1,\ldots,\rho_d^r)$ TO BE DIVERTED TO THE ALTERNATIVE ROUTE
1209
COMPUTE REMAINING TRAFFIC TO BE DIVERTED FROM THE FAILED LINK
$\underline{\rho_f} \leftarrow \underline{\rho_f} - \underline{\rho_d}$

21/3,K/1 (Item 1 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00504439 **Image available**
SYSTEM AND METHOD FOR IMPLEMENTING ERROR DETECTION AND RECOVERY IN A SYSTEM AREA NETWORK
SYSTEME ET PROCEDE SERVANT A METTRE EN OEUVRE UNE DETECTION ET UNE RECUPERATION D'ERREURS DANS UN RESEAU DE SYSTEMES (SAN)
Patent Applicant/Assignee:
TANDEM COMPUTERS INCORPORATED,
Inventor(s):
GARCIA David J,
LARSON Richard O,
LOW Stephen G,
WATSON William J,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9935791 A1 19990715
Application: WO 99US249 19990106 (PCT/WO US9900249)
Priority Application: US 9870650 19980107; US 98224115 19981230
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CA JP AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 7255

Fulltext Availability:
Detailed Description
Claims

English Abstract
...packet reception in a SAN includes requestor and responder nodes, coupled by a plurality of **paths** , that maintain the good and bad status of each **path** and also maintain local copies of a message sequence number. If an error occurs for a **transaction** over a given **path** , the requestor informs the responder, over a good **path** , that the given **path** has failed and both nodes update their **path** status to indicate that the given **path** is bad. A barrier **transaction** is used by the requestor to determine whether the error is transient or permanent, and, if the error is transient, the requestor retries the **transaction**.

French Abstract
...un numero d'ordre de sequence de messages. Si une erreur se produit pour une **transaction** sur un chemin donne, le demandeur informe le repondeur, sur un bon chemin, que le...
...a jour leur etat de chemin pour indiquer que le chemin donne est mauvais. Une **transaction** barriere est utilisee par le demandeur pour determiner si l'erreur est temporaire ou permanente, et, si l'erreur est temporaire, le demandeur essaie de relancer la **transaction**.

Detailed Description
... sender directs data to a particular location in memory utilizing remote direct memory access (RDMA) **transactions** . The initiator of the data transfer specifies both the source buffer and destination buffer of...

...transacti@ns. Furthermore, error recovery should not increase the complexity for the consumer of VIA **services** .

SUMMARY OF THE INVENTION

According to one aspect of the present invention, a SAN maintains local I 0 copies of a sequence number for each data transfer **transaction** at the requestor and responder nodes. Each data transfer is implemented by the SAN as...

...at the requesting node. The responder and requestor nodes are coupled by a plurality of **paths** and each node maintains a record of the good or bad status 1 5 of each **path** . If a **transaction** fails and the **path** is permanently bad both nodes update their status to indicate that the **path** is bad to prevent further **transactions** from including any stale requests potentially still in the network, from ...corrupting data.

According to another aspect of the invention, if an error occurs on a **path** the requestor node implements a barrier **transaction** on the **path** to determine if the failure is permanent or transient.

According to another aspect of the invention, the barrier **transaction** is
perfon-ned by writing a number chosen from a large number space in a...

...3 are block diagrams depicting SAN topologies;
Fig. 4 is a schematic diagram depicting logical **paths** between end nodes of
a SAN;
Fig. 5 is a schematic diagram depicting routers and **links** **connecting** SAN
1 5 end nodes;
Fig. 6 is a graph depicting the transmission of request...
...diagrams showing the state that software on
the requestor and responder moves through for each **path** ;
Fig. 8 is a graph depicting retransmission during error recovery due to a lost request...support one or two ports, each with its AC (media access) and physical layer.

associated **transaction** , packet, **link** -level, M
Similarly, routing nodes with a common routing layer may support multiple ports, each with its associated **link** -level, MAC and physical layer.

Support for two ports enables ServerNet 11 SAN to be...

...Figure 3. On a fault tolerant network, a port of each end node may be **connected** to each network to provide continued VI message communication in the event of failure of...

...into pairs to provide duplex FT controllers. The fabric is the collection of routers, switches, **connectors** , and cables that **connects** the nodes in a network.

The following describes general ServerNet II terminology and concepts. The...

...a packet frarning command. Other commands, used for flow control, virtual channel SUppO113 and other **link** management **functions** , may be embedded within a packet. Each request or response packet defines a variety of information for routing, **transaction** type, verification, length and VI specific information.

I. Routing in the ServerNet 11 SAN is...

...node port in the network is uniquely defined by a 20 bit Port SNID (ServerNet **Node** ID). The **first** 3 bytes of a packet contain the Destination port's SNID or DID (destination port...

...Bits (ACB) field and -the fabric ID bit. The ACB is used to specify the **path** (deten-ninistic or **link** -set adaptive) used to route the packet to its destination port as described in the following section.

ii. The **transaction** type fields define the type of session layer operation that this ServerNet 11 packet is...

...an Ack (acknowledgment) or a Nack (negative acknowledgment). the ServerNet 11 SAN also supports other **transaction** types.

ill. **Transaction** verification fields include the source port ID (SID) and a **Transaction** Serial Number. The **transaction** serial number enables a port with multiple requests I 0 outstanding to uniquely match responses...

...and the Virtual Interface ID number. The VI Operation field defines the type of VI **transaction** being sent (Send, RDMA Read, RDMA Write) and other control information such as whether the...

...first or last packet in a session layer multi-packet transfer. Based on the VI **transaction** type and control inforination, a 32 bit Immediate data field or a 64 bit Virtual...

...a pad byte.

vii. The CRC field contains a checksum computed over the entire packet.

Transaction Overview
The basic flow of transactioils through the ServerNet 11 SAN will ilow be described. VI requires the support of Send, RDMA read and RDMA write **transactions** .

These are translated by the VI session layer into a set of ServerNet 11 **transactions** (request/response packet pairs). All data transfers (e.g., reading a disk file to CPU...

...dumping large volumes of data from a disk farm directly over a high-speed communications **link** , one end node simply interrupting another) consist of one or more such **transactions** .

Creating a Request Packet
The VI User Agent provides the low level routines for VIA...

...the ports simultaneously. This latter feature is called Multi-pathing.

ServerNet 11 end nodes can **connect** both their ports to a single network fabric so that there are up to four possible **paths** between ServerNet II end nodes. Each 1 5 port of a single end node may have a unique ServerNet ID (SNID). Fig. 4 depicts the four possible **paths** that End node A can use when sending request to End node B.

1) End...to End node B SNID [I]
Fig. 5 depicts a network topology utilizing routers and **links** . In Fig. 5, end nodes A-F, each having first and second send receive ports 0 and 1, are coupled by a ServerNet topology including routers RI-R4. **Links** are represented by lines coupling ports to routers or routers to routers.

A first adaptive...

...adaptive set (fat pipe) 4 couples routers R2 and R4.

Routing may be deterministic or **link** set adaptive.'An adaptive **link** -set is Li set of **links** (also called lanes) between two routers that have been grouped to provide higher bandwidth. The...

...source port to a destination port. In deterministic routing the ACB field selects a single **path** or lane through an adaptive **link** -set. Send **transactions** for a particular VI require strict ordering and;therefore use deterministic routing.

RDMA **transactions** , on the other hand, may make use of all possible **paths**
in the network without regard for the ordering of packets within the **transaction** . These **transactions** may use **link** -set adaptive routing as described below. The ACB field specifies which specific **link** (or lane) in this **link** -set is to be used for deterministic routing.

Alternatively, the ACB field can specify **link** -set adaptivity which enables the packets to dynamically choose from any of the **links** in the **link** -set.

A sample topology with several different examples of multipathing using lirik and **path** adaptivity is shown in figure 5.

1 0 Multipathing allows large block transfers done with RDMA Read or Write operations to simultaneously use both ports as well as adaptive **links** between the two communicating NICs. Since the data transfer characteristics of any one VI are...

...VI. Sends from one VI must be sent strictly ordered. Since there are no ordering **guarantees** between packets originating from different ports on a NIC, only one port may be used per Send. Furthermore, only a single ordered **path** through the Network may be used, as described in the following.

Transaction and Packet Layers
The **transaction** layer builds the ServerNet 11 request packet by filling in the appropriate SID, **Transaction** Serial Number (TSN), and CRC. The SID assigned to a packet always corresponds to the...to be checked at the end node and by routers enroute.

Following the ServerNet 11 **link** protocol, the packet is encoded in a series of data symbols followed by a status command. The ServerNet II **link** layer uses other commands for flow control and **link** management. These commands may be inserted anywhere in the **link** data stream, including between consecutive data symbols of a packet. Finally, the symbols are passed...

...for transmission on the physical media to an intermediate routing node.

Routing
The routing control **function** is programmable so that the packet routing can be changed as needed when the network...

...Router nodes serve as crossbar switches; a packet on any incoming (receive) side of a **link** can be switched to the outgoing (transmit) side of any **link** . As the incoming request packet arrives at a router

node , the **first** three bytes, containing the DID, and ACB fields, are decoded and used to select a **link** leading to the destination node. If the transmit side of the selected **link** is not **busy** , the head of the packet is sent to the destination node whether or not the tail of the packet has arrived at the routing node. If the selected **link** is I 0 **busy** with another packet, the newly arrived packet must wait for the target port to become...

...and updates the-packet status (good or bad). The packet status is carried by a **link** symbol TPG (this packet good) or TPB (this packet bad) appended at the end of the packet. Since 1 5 packet status is checked on each **link** , a packet status transition (good to bad) can be attributed to a specific **link** . The packet routing process described above is repeated for each router node in the selected **path** to the destination node.

Receiving a Request Packet
When the request packet arrives at the...successful read request, for example, would include the read data in the ACK response. The **source node** ID from the request packet is used as the destination node ID for the response packet. The response packet must be returned to the original source port. The **path** taken by the response is not necessarily the reverse of the **path** taken by the request. The network may be configured so that responses take very different **paths** than requests. If strict ordering is not required, the response, like the request, may U'se **link** -set adaptivity. The response packet is routed back to the SNID specified by the SID...

...the TSN and the
packet validity checks. If an ACK response passes these tests, the **transaction** layer passes I 0 the response data to the session layer, frees resources associated with the request, and reports the **transaction** as complete. If a NACK response passes these tests, the end node reports the failure of the **transaction** to the session layer. If a valid ACK/NACK response is not received within the...

...acknowledgments to unordered packets before starting the next descliptor.

Ordering of Send Packets Presented to **Transaction** Layer
The VI architecture has no explicit ordering rules as to how the packets that make up a single descriptor are ordered among themselves. That is, VIA only **guarantees** the message ord.eYing the client will see. For example, VIA requires that Send descriptors...order. As long as deterministic routing is used, the networi assures strict ordering along a **path** from a particular **source node** to a particular destination node. This is necessary because the receiving node places the incoming...

...worry about the network providing strict ordering and can choose an arbitrary source port, adaptive **link** set, and destination port for each message.

b. The end node can restrict all the...

...VI to use the same source port, the same destination port, and a single adaptive **path** . By choosing only one **path** through the network, the end node is **guaranteed** that each Send packet it launches into the network will arrive at the destination in...

...node to maintain state per VI

that indicates which source port destination port and adaptive **path** is currently in use for that particular VI. Furthermore, the second approach allows the hardware...

...packets. The end node is free to use different source ports, destination ports and adaptive 1paths for the packets. This freedom can be exploited for a performance gain through multipathing; simultaneously sending the RDMA packets of a single message across multiple **paths** .
When RDMA Read or Write packets are sent over a **path** that does not exhibit strict ordering with the Send packets from the same VI, care...

...the VI is created, the requestor on one node and the responder on the remote **node** **initialize** their sequence numbers to a common value, zero in the preferred embodiment.

After this, the...all other checks are passed, the packet is Acknowledged and committed to memory. If the **transaction** is ordered then the responder then increments its sequence number. If the **transaction** is unordered than the responder
does not increment its sequence number;
an out-of-sequence...

...unordered packets is given in Fig. 6. In Fig. 6, during the first two Send **transactions** , the responder checks that the SEQ in the packet matches the local copy of Rsp...

...is incremented after each response packet is transmitted. At end of the first two Send **transactions** , Rqst. SN and Rsp. SN both equal 6. The packets for the RDMA include an...
...or responder increments its local copy of SN. Thus, at the end of the RDMA **transaction** both Rqst. SN and Rsp. SN = 6. The first packet of the subsequent Send **transaction** has SEQ = 6 and SEQ matches the local copy of Rsp. SN. Since Send packets...matches incoming responses with the originating
request by comparing the Sourcell), VI number, Sequence number, **transaction** type, Lind **Transaction** Serial Number (TSN) with that of the originating request.

Error Recovery and **Path** State
Error recovery is **initiated** by the requesting **node** whenever [lie requestor fails to get a positive acknowledgment for each of its request packets...

...operation(s) to flush out any errant request or
response packets.

2) Disabling A bad **path** if the barrier operation failed.

) Retransmitting from the earliest packet that had failed.

The first...
...diagrams showing the state that software on the requestor and responder moves through for each **path** . In Fig. 7, dashed lines represent Kernel to Kernel Supervisory Proto6o-I messages that modify the remote node's state.

The ServerNet architecture allows multiple **paths** between end nodes. The requestor repeats these two basic steps on each **path** until the packet is transmitted successfully.

The requestor and responder SW each maintain a view of the state of each I 0 **path** . The requestor uses its view of the **path** state to determine which **path** it uses for Send and RDMA operations. The responder uses its view of the **path** state to determine which input **paths** it allows incoming requests on. The responder logic maintains a four bit field (ReqIn PathVector...

...VI in use. Each of the four bits corresponds to one of the four possible **paths** between the requestor's two ports and the responder's two ports.

1 5 The...

...set
The requestor and responder communicate using the kernel-to-kernel Supervisory protocol to communicate **path** state changes.

The requestor's view of the **path** state transitions from good to bad whenever the requestor fails to get an acknowledgment (either...

...by getting a time-out error.

The requestor can attempt a barrier operation on the **path** to see if the failure is perri-lanent or transient. If the barrier succeeds, the **path** is considered good and the original operation can be retried. If the barrier falls, the requestor must resort to a different good **path** .

Before the requestor can try a different good **path** , the requestor must inforrn the destination that the original **path** is bad. This is done, by any **path** possible.

For example, in VIA the Kernel Agent to Kernel Agent Supervisory Protocol is used.

After the destination is informed the **path** is bad the destination disables a bit in a four bit field (ReqlnPath Vector), thereby ignoring incoming requests from that **path** . The requestor then stops using the bad **path** until a subsequent barrier **transaction** determines that the **path** is good. After the destination acknowledges the supervisory protocol message, indicating that the destination has disabled requests from the offending **path** , the requestor is free to retry the message on a different **path** .

After a time-out error, the requestor attempts to bring the **path** back to a useful state by completing a barrier operation. The barrier operation ensures there...

...in error recovery to flush any stale request or stale response packets from a particular **path** in the SAN. A **path** is the collection of ServerNet **links** between a specific port of two end nodes.

A VIA barrier operation is done with...

...If the read value matches the write value, then the barrier succeeded and there are **guaranteed** to be no more Send or RDMA request or response packets on that **path** between the requestor and responder.

If the RDMA operation fails because the number read back...
...and fulfilled the barrier.

Note that the barrier needs to be done separately on all **paths** the RDMA

operation could have taken. That is, if the RDMA operation was being generated from multiple source ports (multipathing) and was using full **link** adaptivity (the packets were allowed to take any one of four possible "lanes"), then separate...

...be done from each source port to each destination port, over each of the possible **link** adaptive **paths** .

The barrier operatioii must be done for each of the possible "lalles" between a specific...

...destination port have no remaining request or response packets lurking in the SAN.

If a **path** traverses a "fat-pipe" a separate barrier must be sent down each lane of the...

...between any given source/destination pair. The barrier need only be sent along the same **path** as the original request that failed. There is no requirement for the barrier to be...SEQ = 1 is corrupted. The missing response is detected when the requestor times out its **transaction** . The requestor resets in send engine to start generating packets at the one that failed...

...them and throws the data away.

Note that the response packets for this particular RDMA **transaction** all have the same value of SEQ because the request SNs and response SNs are not incremented for RDMA **transactions** that are unordered. In this case the TSNs are utilized by requestor to match response...

Claim

... data between a requestor node and a responder node, wit h the requestor and responder **nodes** coupled by **first** and second **paths** and with the SAN implementing **data** transfers as a **sequence** of request/response packet pairs, and with the SAN for implementing ordered **transactions** requiring that packets be received in the order transmitted and remote direct memory access packets...

...the steps of-. maintaining, at said requestor, the request out status of each of said **paths** as good or bad, so that only good **paths** are utilized for data transfer **transactions** ; 1 1 maintaining, at said responder, a request in status of each of said **paths** as good or bad, so that requests are accepted only on good **paths** ; detecting, as said requestor, if the first **path** falls, and implementing a barrier **transaction** , to determine whether said failure is transient or pennanent; 1 5 if transient, at said requestor, retrying said **transaction** on said first **path** ; if permanent, at said requestor, utilizing the second **path** to infon-n said 1 7 responder that the first **path** is bad; 1 8 at said responder; updating the responder request in status of the first **path** 1 9 to indicate that the first **path** is bad so that packets will not be accepted from said first **path** ; at the requestor, updating the request out status of the first **path** to indicate that the first **path** is bad so that subsequent **transaction** do not use said first **path** .

2 The method of claim I wherein said step of detecting comprises: ? at the requestor;, maintaining a request sequence number and ail

elapsecl time since the beginning of a **transaction** including the request sequence number as a packet sequence number in a request packet and, for an ordered **transaction** , incrementincy
CD
the request sequence number after the request packet is sent;
at the responder...

...only if the packet sequence number matches the responder
I 0 sequence number and the **transaction** is an ordered **transaction** ; and
I I at the requestor: indicating that the first **path** is bad if, after a fixed time has elapsed, an acknowledge packet has not been...

...sequence number.
3 The method of claim 1 wherein said step of implementing a barrier **transaction** in order to verify the correct operation of **path** and to ensure there are no
remaining request or response packets in SAN, comprises the...

...of:
at the requestor, implementing a remote direct access memory write operation along said first **path** to write an arbitrary ...transferring data between a requestor node and a responder node, with the requestor and responder **nodes** coupled by **first** and second **paths** and with the SAN implementing **data** transfers as a **sequence** of request/response packet pairs, and with the SAN for implementing ordered **transactions** requiring that packets be received in the order transmitted and remote direct memory access packets...

...said 1 2 requestor status logic maintaining the request out status of each of said **paths** as good oil 3 bad, so that only good **paths** are utilized for data transfer **transactions** , and, if the first **path** 1 4 is determined to be bad, Updating the request out status Of the first **path** to indicate that the 1 5 first **path** is bad so that subsequent **transaction** do not use said first **path** and with the
1 6 kernel agent software:
detecting, as said requestor, if the first **path** fails, and implementing
1 8 a barrier **transaction** , to detenninie whether said failure is transient or permanent;
if transient, at said requestor, retrying said **transaction** on said first
path ;
if permanent, at said requestor, utilizing the second **path** to inforin said responder that the first **path** is bad; and with the SAN comprising:
a responder end node including a responder network...

...logic, with said responder status logic maintaining a request in status of each of said **paths** as good or bad, so that requests are accepted only on good **paths** , and, if the first **path** is determined to be bad, updating the responder request in status of the first **path** to indicate that the first **path** is bad so that packets will not be accepted from said first **path** .

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :		(11) International Publication Number:	WO 99/35791
H04L 12/56, 12/44, 29/14, 12/26, 29/06, 1/18	A1	(43) International Publication Date:	15 July 1999 (15.07.99)

(21) International Application Number: PCT/US99/00249

(22) International Filing Date: 6 January 1999 (06.01.99)

(30) Priority Data:

60/070,650	7 January 1998 (07.01.98)	US
09/224,115	30 December 1998 (30.12.98)	US

(71) Applicant: TANDEM COMPUTERS INCORPORATED [US/US]; 10435 North Tantau Avenue, Loc. 200-16, Cupertino, CA 95014-0709 (US).

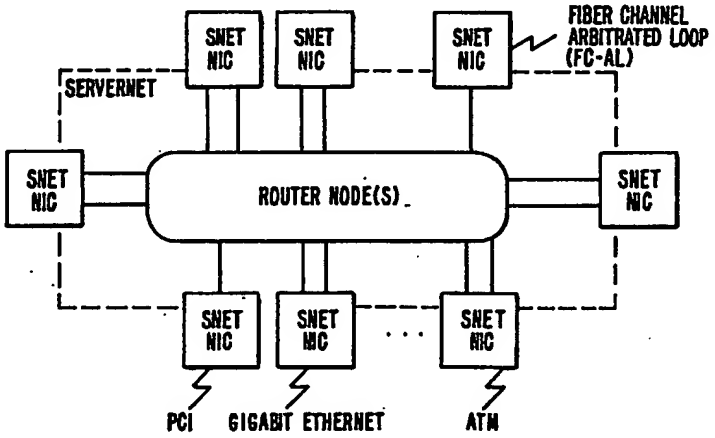
(72) Inventors: GARCIA, David, J.; 24100 Hutchinson Road, Los Gatos, CA 95033 (US). LARSON, Richard, O.LOW, Stephen, G.; 4301 Avenue D., Austin, TX 78751 (US). WATSON, William, J.; 1501 Ulrich Avenue, Austin, TX 78756 (US).

(74) Agents: KRUEGER, Charles, E. et al.; Townsend and Townsend and Crew LLP, 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR IMPLEMENTING ERROR DETECTION AND RECOVERY IN A SYSTEM AREA NETWORK



(57) Abstract

A system and method for facilitating both in-order and out-of-order packet reception in a SAN includes requestor and responder nodes, coupled by a plurality of paths, that maintain the good and bad status of each path and also maintain local copies of a message sequence number. If an error occurs for a transaction over a given path, the requestor informs the responder, over a good path, that the given path has failed and both nodes update their path status to indicate that the given path is bad. A barrier transaction is used by the requestor to determine whether the error is transient or permanent, and, if the error is transient, the requestor retries the transaction.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM AND METHOD FOR IMPLEMENTING ERROR DETECTION AND RECOVERY IN A SYSTEM AREA NETWORK

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a claims priority from Provisional Appln. No. 60/070,650, filed January 7, 1998, the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

Traditional network systems utilize either channel semantics (send/receive) or memory semantics (DMA) model. Channel semantics tends to be used in I/O environments and memory semantics in processor environments.

In the channel semantics model, the sender does not know where data is to be stored, it just puts the data on the channel. On the sending side, the sending process specifies the memory regions that contain the data to be sent. On the receiving side, the receiving process specifies the memory regions where the data will be stored.

In the memory semantics model, the sender directs data to a particular location in memory utilizing remote direct memory access (RDMA) transactions. The initiator of the data transfer specifies both the source buffer and destination buffer of the data transfer. There are two types of RDMA operations, read and write.

The virtual interface architecture (VIA) has been jointly developed by a number of computer and software companies. VIA provides consumer processes with a protected, directly accessible interface to network hardware, termed a virtual interface. VIA is especially designed to provide low latency message communication over a system area network (SAN) to facilitate multi- processing utilizing clusters of processors.

A SAN is used to interconnect nodes within a distributed computer system, such as a cluster. The SAN is a type of network that provides high bandwidth, low latency communication with a very low error rate. SANs often utilize fault-tolerant

It is important for the SAN to provide high reliability and high-bandwidth, low latency communication to fulfill the goals of the VIA. Further, it is important for the SAN to be able to recover from errors and continue to operate in the event of equipment failures. Error recovery must be accomplished without high CPU overhead associated with all transactions. Furthermore, error recovery should not increase the complexity for the consumer of VIA services.

SUMMARY OF THE INVENTION

According to one aspect of the present invention, a SAN maintains local copies of a sequence number for each data transfer transaction at the requestor and responder nodes. Each data transfer is implemented by the SAN as a sequence of request/response packet pairs. An error condition arises if a response to any request packet is not received at the requesting node. The responder and requestor nodes are coupled by a plurality of paths and each node maintains a record of the good or bad status of each path. If a transaction fails and the path is permanently bad both nodes update their status to indicate that the path is bad to prevent further transactions from including any stale requests potentially still in the network, from arriving at the destination and potentially corrupting data.

According to another aspect of the invention, if an error occurs on a path the requestor node implements a barrier transaction on the path to determine if the failure is permanent or transient.

According to another aspect of the invention, the barrier transaction is performed by writing a number chosen from a large number space in a way that minimizes the probability of reusing the number in a short period of time.

According to one aspect of the invention, the number is randomly chosen from a large number space.

According to another aspect of the invention, the large number is based on the requestor ID and an incrementing component managed by the requestor.

According to another aspect of the invention, if the failure is transient the requestor retransmits packets starting with the packet that first caused an error condition to be detected.

According to another aspect of the invention, a sequence number is included in each request packet and copied into each response packet. A local copy of the

sequence number is maintained at the requestor and responder. If the sequence number in the request packet does not match the sequence number at the responder a negative acknowledge response packet is generated.

Other features and advantages of the invention will be apparent in view of the following detailed description and appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram depicting ServerNet protocol layers implemented by hardware, where ServerNet is a SAN manufactured by the assignee of the present invention;

Figs. 2 and 3 are block diagrams depicting SAN topologies;

Fig. 4 is a schematic diagram depicting logical paths between end nodes of a SAN;

Fig. 5 is a schematic diagram depicting routers and links connecting SAN end nodes;

Fig. 6 is a graph depicting the transmission of request and response packets between a requestor and a responder end node. Fig. 6 Shows the sequence numbers used in packets for three Send operations, an RDMA operation, and two additional Send operations. The diagram shows the sequence numbers maintained in the requestor logic, the sequence number contained in each packet, and the sequence numbers maintained at the responder logic;

Fig. 7 is two interlocked state diagrams showing the state that software on the requestor and responder moves through for each path;

Fig. 8 is a graph depicting retransmission during error recovery due to a lost request packet; and

Fig. 9 is a graph depicting retransmission during error recovery due to a lost acknowledgment packet.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

The preferred embodiments will be described implemented in the ServerNet II (ServerNet) architecture, manufactured by the assignee of the present invention, which is a layered transport protocol for a System Area Network (SAN) optimized to support the Virtual Interface (VI) architecture session layer which has

stringent user-space to user-space latency and bandwidth requirements. These requirements mandate a reliable hardware (HW) message transport solution with minimal software (SW) protocol stack overhead. The ServerNet II protocol layers for an end node VI Network Interface controller/Card (NIC) and for a routing node are illustrated in Figure 1. A single NIC and VI session layer may support one or two ports, each with its associated transaction, packet, link-level, MAC (media access) and physical layer. Similarly, routing nodes with a common routing layer may support multiple ports, each with its associated link-level, MAC and physical layer.

Support for two ports enables ServerNet II SAN to be configured in both non-redundant and redundant (fault tolerant, or FT) SAN configurations as illustrated in Figure 2 and Figure 3. On a fault tolerant network, a port of each end node may be connected to each network to provide continued VI message communication in the event of failure of one of the SANs. In the fault tolerant SAN, nodes may be also ported into a single fabric or single ported end nodes may be grouped into pairs to provide duplex FT controllers. The fabric is the collection of routers, switches, connectors, and cables that connects the nodes in a network.

The following describes general ServerNet II terminology and concepts. The use of the term "layer" in the following description is intended to describe functionality and does not imply gate level partitioning.

Two ports are supported on a NIC for both performance and fault tolerance reasons. Both of these ports operate under the same session layer VIA engine. That is, data may arrive on any port and be destined for any VI. Similarly, the VIs on the end node can generate data for any of these ports.

ServerNet II packets are comprised of a series of data symbols followed by a packet framing command. Other commands, used for flow control, virtual channel support, and other link management functions, may be embedded within a packet. Each request or response packet defines a variety of information for routing, transaction type, verification, length and VI specific information.

i. Routing in the ServerNet II SAN is destination based using the first 3 bytes of the packet. Each NIC end node port in the network is uniquely defined by a 20 bit Port SNID (ServerNet Node ID). The first 3 bytes of a packet contain the Destination port's SNID or DID (destination port ID) field, a three bit Adaptive Control Bits (ACB) field and the fabric ID bit. The ACB is used to specify the path (deterministic

or link-set adaptive) used to route the packet to its destination port as described in the following section.

ii. The transaction type fields define the type of session layer operation that this ServerNet II packet is carrying and other information such as whether it is a request or a response and, if a response, whether it is an Ack (acknowledgment) or a Nack (negative acknowledgment). the ServerNet II SAN also supports other transaction types.

iii. Transaction verification fields include the source port ID (SID) and a Transaction Serial Number. The transaction serial number enables a port with multiple requests outstanding to uniquely match responses to requests.

iv. The Length field consists of an encoding of the number of bytes of payload data in the packet. Payloads up to 512 bytes are supported and code space is reserved for future increases in payload size.

v. The VI Session Layer specific fields describe VI information such as the VI Operation, the VIA Sequence number, and the Virtual Interface ID number. The VI Operation field defines the type of VI transaction being sent (Send, RDMA Read, RDMA Write) and other control information such as whether the packet is ordered or unordered, whether there is immediate data and/or whether this is the first or last packet in a session layer multi-packet transfer. Based on the VI transaction type and control information, a 32 bit Immediate data field or a 64 bit Virtual address may follow the VI ID number.

vi. The payload data field carries up to 512 bytes of data between requestors and responders and may contain a pad byte.

vii. The CRC field contains a checksum computed over the entire packet.

Transaction Overview

The basic flow of transactions through the ServerNet II SAN will now be described. VI requires the support of Send, RDMA read and RDMA write transactions. These are translated by the VI session layer into a set of ServerNet II transactions (request/response packet pairs). All data transfers (e.g., reading a disk file to CPU memory, dumping large volumes of data from a disk farm directly over a high-speed communications link, one end node simply interrupting another) consist of one or more such transactions.

Creating a Request Packet

The VI User Agent provides the low level routines for VIA Send, RDMA Write, and RDMA Read operations. These routines place a descriptor for the desired transfer in the appropriate VI queue and notify the VIA hardware that the descriptor is ready for processing. The VIA hardware reads the descriptor, and based on the descriptor contents, builds the ServerNet request packet header and assembles the data payload (if appropriate).

Dual Ports and Ordering

In a NIC with two ports, it is possible for a single VIA interface to process Sends and RDMA operations from several different VIs in parallel. It is also possible for a large RDMA transfer from a single VI to be transferred on both of the ports simultaneously. This latter feature is called Multi-pathing.

ServerNet II end nodes can connect both their ports to a single network fabric so that there are up to four possible paths between ServerNet II end nodes. Each port of a single end node may have a unique ServerNet ID (SNID). Fig. 4 depicts the four possible paths that End node A can use when sending request to End node B:

1) End node A SNID[0] to End node B SNID[0]
2) End node A SNID[0] to End node B SNID[1]
3) End node A SNID[1] to End node B SNID[0]
4) End node A SNID[1] to End node B SNID[1]

Fig. 5 depicts a network topology utilizing routers and links. In Fig. 5, end nodes A-F, each having first and second send receive ports 0 and 1, are coupled by a ServerNet topology including routers R1-R4. Links are represented by lines coupling ports to routers or routers to routers. A first adaptive set (fat pipe) 2 couples routers R1 and R3 and a second adaptive set (fat pipe) 4 couples routers R2 and R4.

Routing may be deterministic or link set adaptive. An adaptive link-set is a set of links (also called lanes) between two routers that have been grouped to provide higher bandwidth. The Adaptive Control Bits (ACB) specify which type of routing is in effect for a particular packet.

Deterministic routing preserves strict ordering for packets sent from a particular source port to a destination port. In deterministic routing the ACB field selects a single path or lane through an adaptive link-set. Send transactions for a particular VI require strict ordering and therefore use deterministic routing.

RDMA transactions, on the other hand, may make use of all possible paths in the network without regard for the ordering of packets within the transaction. These transactions may use link-set adaptive routing as described below. The ACB field specifies which specific link (or lane) in this link-set is to be used for deterministic routing.

Alternatively, the ACB field can specify link-set adaptivity which enables the packets to dynamically choose from any of the links in the link-set.

A sample topology with several different examples of multipathing using link and path adaptivity is shown in figure 5.

Multipathing allows large block transfers done with RDMA Read or Write operations to simultaneously use both ports as well as adaptive links between the two communicating NICs. Since the data transfer characteristics of any one VI are expected to be bursty, multipathing allows the end node to marshal all its resources for a single transfer. Note that multipathing does not increase the throughput of multiple Send operations from one VI. Sends from one VI must be sent strictly ordered. Since there are no ordering guarantees between packets originating from different ports on a NIC, only one port may be used per Send. Furthermore, only a single ordered path through the Network may be used, as described in the following.

Transaction and Packet Layers

The transaction layer builds the ServerNet II request packet by filling in the appropriate SID, Transaction Serial Number (TSN), and CRC. The SID assigned to a packet always corresponds to the SNID of the port the packet originates from. The TSN can be used to help the port manage multiple outstanding requests and match the resulting responses uniquely to the appropriate request. The CRC enables the data integrity of the packet to be checked at the end node and by routers enroute.

Following the ServerNet II link protocol, the packet is encoded in a series of data symbols followed by a status command. The ServerNet II link layer uses other commands for flow control and link management. These commands may be inserted anywhere in the link data stream, including between consecutive data symbols of a packet. Finally, the symbols are passed through the MAC layer for transmission on the physical media to an intermediate routing node.

Routing

The routing control function is programmable so that the packet routing can be changed as needed when the network configuration changes (e.g., route to new end nodes). Router nodes serve as crossbar switches; a packet on any incoming (receive) side of a link can be switched to the outgoing (transmit) side of any link. As the incoming request packet arrives at a router node, the first three bytes, containing the DID, and ACB fields, are decoded and used to select a link leading to the destination node. If the transmit side of the selected link is not busy, the head of the packet is sent to the destination node whether or not the tail of the packet has arrived at the routing node. If the selected link is busy with another packet, the newly arrived packet must wait for the target port to become free before it can pass through the crossbar.

As the tail of the packet arrives, the router node checks the packet CRC and updates the packet status (good or bad). The packet status is carried by a link symbol TPG (this packet good) or TPB (this packet bad) appended at the end of the packet. Since packet status is checked on each link, a packet status transition (good to bad) can be attributed to a specific link. The packet routing process described above is repeated for each router node in the selected path to the destination node.

Receiving a Request Packet

When the request packet arrives at the destination node, the ServerNet II interface receiver checks its validity (e.g., must contain correct destination node ID, the length is correct, the Fabric bit in the packet matches the Fabric bit associated with the receiving port, the request field encodes a valid request, and CRC must be good,). If the packet is invalid for any reason, the packet is discarded. The ServerNet II interface may save error status for evaluation by software. If these validity checks succeed, several more checks are made. Specifically, if the request specifies an RDMA Read or Write, the address is checked to ensure access has been enabled for that particular VI. Also, the input port and Source ID of the packet are checked to ensure access to the particular VI is allowed on that input port from the particular Source. If the packet is valid, the request can be completed.

Response Packet

A response is created based on the success (ACK response) or failure (NAK response) of the request packet. A successful read request, for example, would include the read data in the ACK response. The source node ID from the request packet is

used as the destination node ID for the response packet. The response packet must be returned to the original source port. The path taken by the response is not necessarily the reverse of the path taken by the request. The network may be configured so that responses take very different paths than requests. If strict ordering is not required, the response, like the request, may use link-set adaptivity. The response packet is routed back to the SNID specified by the SID field of the request. The ACB field of the request packet is also duplicated for the response packet.

The response can be matched with the request using the TSN and the packet validity checks. If an ACK response passes these tests, the transaction layer passes the response data to the session layer, frees resources associated with the request, and reports the transaction as complete. If a NACK response passes these tests, the end node reports the failure of the transaction to the session layer. If a valid ACK/NACK response is not received within the allotted time limit, a time-out error is reported.

The requestor can stream many strictly ordered ServerNet II messages onto the wire before receiving an acknowledgment. The sliding window protocol allows the requestor to have up to 128 packets outstanding per VI.

The hardware can operate in one of two modes with respect to generating multiple outstanding request packets:

1. The hardware can stream packets from the same VI send queue onto the wire, and start the next descriptor before receiving all the acknowledgments from the current descriptor. This is referred to as "Next Descriptor After Launch" or NDAL.

2. The hardware can stream packets to a single descriptor onto the wire but wait for all the outstanding acknowledgments to complete before starting the next descriptor. This is referred to as "Next Descriptor after Ack" or NDAA.

The choice of NDAL or NDAA modes of operation is determined by how strongly ordered the packets are generated.

Ordered and unordered messages may be mixed on a single VI. When generating an unordered message, the requestor must wait for completion of all acknowledgments to unordered packets before starting the next descriptor.

Ordering of Send Packets Presented to Transaction Layer

The VI architecture has no explicit ordering rules as to how the packets that make up a single descriptor are ordered among themselves. That is, VIA only guarantees the message ordering the client will see. For example, VIA requires that Send

descriptors for a particular VI be completed in order, but the VIA specification doesn't say how the packets will proceed on the wire.

The ServerNet II SAN requires that all Send packets destined for a particular VI be delivered by the SAN in strict order. As long as deterministic routing is used, the network assures strict ordering along a path from a particular source node to a particular destination node. This is necessary because the receiving node places the incoming packets into a scatter list. Each incoming packet goes to a destination determined by the sum total of bytes of the previous packets. The strict ordering of packets is necessary to preserve integrity of the entire block of data being transferred because incoming packets are placed in consecutive locations within the block of data. Each packet has a sequence number to allow the receiver to detect an out of order, missing, or repeated packet.

There are two ways for an end node to meet these ordering requirements:

a. The end node can wait for the acknowledgment from each Send packet to complete before starting another Send packet for that VI. By waiting for each acknowledgment the end node doesn't have to worry about the network providing strict ordering and can choose an arbitrary source port, adaptive link set, and destination port for each message.

b. The end node can restrict all the Send operations for a given VI to use the same source port, the same destination port, and a single adaptive path. By choosing only one path through the network, the end node is guaranteed that each Send packet it launches into the network will arrive at the destination in order.

The second approach requires the VIA end node to maintain state per VI that indicates which source port destination port and adaptive path is currently in use for that particular VI. Furthermore, the second approach allows the hardware to process descriptors in the higher performance NDAL mode.

With the second approach, Send packets from a single VI can stream onto the network without waiting for their accompanying acknowledgments. An incrementing sequence number is used so the destination node can detect missing, repeated, or unordered Send packets.

Ordering of RDMA Packets

RDMA operations have slightly different ordering requirements than Send operations. An RDMA packet contains the address to which the destination end node

writes the packet contents. This allows multiple RDMA packets within an RDMA message to complete out of order. The contents of each packet are written to the correct place in the end node's memory, regardless of the order in which they complete.

RDMA request packets may be sent ordered or unordered. A bit in the packet header is set to a 1 for ordered packets and is set to a 0 for unordered packets. As will be explained later, this bit is used by the responder logic to determine if it should increment its copy of the expected sequence number. Sequence numbers do not increment for unordered packets. The end node is free to use different source ports, destination ports and adaptive paths for the packets. This freedom can be exploited for a performance gain through multipathing; simultaneously sending the RDMA packets of a single message across multiple paths.

When RDMA Read or Write packets are sent over a path that does not exhibit strict ordering with the Send packets from the same VI, care must be taken when launching packets for the following message. The next message cannot be started until the last acknowledgment of the RDMA Read or Write operation successfully completes.

In other words, when multipathing is used to generate RDMA Read or Write requests, the hardware must operate in the NDAA mode. This ensures the RDMA Read or Write is completed before moving on to subsequent descriptors.

An end node may choose to send RDMA packets strictly ordered. This can be advantageous for smaller RDMA transfers as the hardware can operate in NDAL mode. The VI can proceed to the next descriptor immediately after launching the last packet of a message that is sent strictly ordered (and hence used incrementing sequence numbers).

Ordering of Generated Response Packets at the Responder

The ServerNet II end node must respond to incoming Send requests and RDMA Write requests from a particular VI in strict order, and must write these packets to memory in strict order.

The ServerNet II end node must also respond to incoming RDMA Read requests from a particular VI in strict order.

Because response packets are transported by the network in strict order, the requestor will receive all incoming response packets for a particular VI in the same order as that in which the corresponding requests were generated.

VIA Message Sequence Numbers

The ServerNet SAN uses acknowledgment packets to inform the requestor that a packet completed successfully. Sequence numbers in the packets (and acknowledgments) are used to allow the sender to support multiple outstanding requests to ensure adequate performance and to be able to recover from errors occurring in the network.

Fig. 6 is a graph depicting the generation, checking, and updating of VIA sequence numbers at requestor and responder nodes. In Fig. 6 time increases in the downward direction. Requests are indicated by solid arrows directed to the right and responses by dotted arrows directed to the left.

Sequence Number Initialization

The requestor and responder logic each maintain an 8 bit sequence number for each VI in use. When the VI is created, the requestor on one node and the responder on the remote node initialize their sequence numbers to a common value, zero in the preferred embodiment.

After this, the requestor places its sequence number into each of the outgoing request packets. As depicted in Fig. 6, the sequence number, SEQ, in included in each request packet. The responder compares the sequence number from the incoming request packet with the responder's local copy. The responder uses this comparison to determine if the packet is valid, if it is a duplicate of a packet already received, or if it is an out-of-sequence packet. An out-of-sequence packet can only happen if the responder missed an incoming packet. The responder can choose to return a 'sequence error NACK packet' or it can simply ignore the out-of-sequence packet. In the latter case, the requestor will have a timeout on the request (and presumably on the packet the responder missed) and initiate error recovery. Generating a Sequence Error NACK Packet is preferred as it forces the requestor to start error recovery more quickly.

The following describes how the sequence numbers are generated and checked.

Generating Sequence Numbers for Request Packets.

When transmitting ordered packets (i.e. transfers are on a specific source port to a specific destination port and the ACB specifies a specific lane) the request sequence number is incremented after each packet is sent. When transmitting unordered

packets (i.e. multipathing is used and/or the ACB bits specify full link set adaptivity) the request sequence number is not incremented after such a packet is sent.

For example, in Fig. 6, during the first two Send transactions, the local copy of the request sequence number is incremented after the packet is sent (Rqst. SN = 0 to 6). For the RDMA operation, which sends 2500 bytes unordered, the requestor does not increment local copy of the request sequence number (Rqst. SN = 6). The requestor does not increment the local copy of the SN until after the first packet of the Send following the RDMA is transmitted.

Send packets are typically sent fully ordered lest the requestor have to wait for an acknowledgment for each packet before proceeding to the next. On the other hand, RDMA packets may be sent either ordered or unordered. To take advantage of multipathing, a requestor must use unordered RDMA packets.

The sender guarantees to never exceed the windowsize number of packets outstanding per VI. If S is the number of bits in the sequence number, then the windowsize is 2**(S-1).

A packet is outstanding until it and all its predecessors are acknowledged. The requestor does not mark a descriptor done until all packets requested by that descriptor are positively acknowledged.

Checking Sequence Numbers on Incoming Request Packets.

The destination node responding to the incoming request packet checks each incoming request packet to verify its sequence number against the responder's local copy.

The responder logic compares its sequence number with the packet's sequence number to determine if the incoming packet is either:

the expected packet it's looking for (i.e., the packet's sequence number is the same as the sequence number maintained by the responder logic), in which case the responder processes the packet and if all other checks are passed, the packet is Acknowledged and committed to memory. If the transaction is ordered then the responder then increments its sequence number. If the transaction is unordered than the responder does not increment its sequence number;

an out-of-sequence packet (which means an earlier incoming packet must have gotten lost), beyond the one it's looking for in which case the responder Nacks the

packet and throws it away. The receive logic in the VI is not stopped and the responder does not increment its sequence number; or

a duplicate packet (which is being resent because the requestor must not have received an earlier ack) in which case the responder Acks the packet and throws it away. If the request had been an RDMA Read, the responder completes the read operation and returns the data with a positive acknowledgment.

An example of the responder checking sequence numbers for ordered and unordered packets is given in Fig. 6. In Fig. 6, during the first two Send transactions, the responder checks that the SEQ in the packet matches the local copy of Rsp. SN. Since the Send packets include ACB indicating ordered then the Rsp. SN is incremented after each response packet is transmitted. At end of the first two Send transactions, Rqst. SN and Rsp. SN both equal 6. The packets for the RDMA include an ACB indicating unordered receipt is allowed. Neither the requestor or responder increments its local copy of SN. Thus, at the end of the RDMA transaction both Rqst. SN and Rsp. SN = 6. The first packet of the subsequent Send transaction has SEQ = 6 and SEQ matches the local copy of Rsp. SN. Since Send packets are ordered the responder increments its local copy of Rsp. SN.

Sequence Numbers on Response Packets.

When generating either a positive or negative acknowledgment, the responder logic copies the incoming sequence number and uses it in the sequence number field of the acknowledgment.

The requestor logic matches incoming responses with the originating request by comparing the SourceID, VI number, Sequence number, transaction type, and Transaction Serial Number (TSN) with that of the originating request.

Error Recovery and Path State

Error recovery is initiated by the requesting node whenever the requestor fails to get a positive acknowledgment for each of its request packets. A time-out or Nack indicating a sequence number error can cause the requestor's Kernel Agent to start error recovery.

Error recovery involves three basic steps:

1) Completing a barrier operation(s) to flush out any errant request or response packets.

2) Disabling a bad path if the barrier operation failed.

3) Retransmitting from the earliest packet that had failed.

The first two steps will now be described with reference to Fig. 7, which is two interlocked state diagrams showing the state that software on the requestor and responder moves through for each path. In Fig. 7, dashed lines represent Kernel to Kernel Supervisory Protocol messages that modify the remote node's state.

The ServerNet architecture allows multiple paths between end nodes. The requestor repeats these two basic steps on each path until the packet is transmitted successfully.

The requestor and responder SW each maintain a view of the state of each path. The requestor uses its view of the path state to determine which path it uses for Send and RDMA operations. The responder uses its view of the path state to determine which input paths it allows incoming requests on. The responder logic maintains a four bit field (ReqIn PathVector) for each VI in use. Each of the four bits corresponds to one of the four possible paths between the requestor's two ports and the responder's two ports. The requestor only accepts incoming requests from a particular source or destination port if the corresponding bit in the ReqInPathVector is set

The requestor and responder communicate using the kernel-to-kernel Supervisory protocol to communicate path state changes.

The requestor's view of the path state transitions from good to bad whenever the requestor fails to get an acknowledgment (either positive or negative) to a request. The requestor detects the lack of an Ack or Nack by getting a time-out error. The requestor can attempt a barrier operation on the path to see if the failure is permanent or transient. If the barrier succeeds, the path is considered good and the original operation can be retried. If the barrier fails, the requestor must resort to a different good path.

Before the requestor can try a different good path, the requestor must inform the destination that the original path is bad. This is done, by any path possible. For example, in VIA the Kernel Agent to Kernel Agent Supervisory Protocol is used. After the destination is informed the path is bad the destination disables a bit in a four bit field (ReqInPath Vector), thereby ignoring incoming requests from that path. The requestor then stops using the bad path until a subsequent barrier transaction determines that the path is good. After the destination acknowledges the supervisory protocol

message, indicating that the destination has disabled requests from the offending path, the requestor is free to retry the message on a different path.

After a time-out error, the requestor attempts to bring the path back to a useful state by completing a barrier operation. The barrier operation ensures there are no other packets in any buffer that might show up later and corrupt the data transfer.

Barrier operations are used in error recovery to flush any stale request or stale response packets from a particular path in the SAN. A path is the collection of ServerNet links between a specific port of two end nodes.

A VIA barrier operation is done with a RDMA Write followed by an RDMA Read. A number chosen from a large number space (either incrementing or pseudo random) is written to a fixed location (e.g. a page number agreed to, a priori, by the kernel agent-to-kernel agent Supervisory Protocol and either a fixed or random offset within the page). The number is then read back with an RDMA read. If the read value matches the write value, then the barrier succeeded and there are guaranteed to be no more Send or RDMA request or response packets on that path between the requestor and responder.

If the RDMA operation fails because the number read back does not match the number written, then the barrier is tried again. This could have happened because a previous response in the network came back and fulfilled the barrier.

Note that the barrier needs to be done separately on all paths the RDMA operation could have taken. That is, if the RDMA operation was being generated from multiple source ports (multipathing) and was using full link adaptivity (the packets were allowed to take any one of four possible "lanes"), then separate barrier operations must be done from each source port to each destination port, over each of the possible link adaptive paths.

The barrier operation must be done for each of the possible "lanes" between a specific Source port and Destination port. A barrier done on one VI ensures that all other VIs using that source port and destination port have no remaining request or response packets lurking in the SAN.

If a path traverses a "fat-pipe" a separate barrier must be sent down each lane of the fat pipe. SW can either blindly send four barrier operations (one for each lane) or it can maintain state telling how many lanes are in use on the fat pipes between any given source/destination pair. The barrier need only be sent along the same path as the

original request that failed. There is no requirement for the barrier to be sent from or to the same VI.

Turning now to the third step, i. e., .retransmitting from the earliest packet that had failed, after notification of the error the requestor retransmits the packets starting at (or before) the packet that failed to receive a positive acknowledgment. The requestor can restart up to WindowSize number of packets. The responder logic acks and then ignores any resent packets if they have already been stored to the receive queue. When the correct packet is reached, the responder logic can tell from the sequence number that it is now time to resume writing the data to the receive queue.

Examples of retransmission after failure to receive a response are depicted in Figs. 8 and 9. In Fig. 8, the request packet with SEQ=2 is corrupted. The missing request is detected by the responder on the next packet and Nacked (Negative Acknowledged) and all subsequent packets are thrown away and Nacked. The requestor resets its send engine to start generating packets at the one that failed to receive an Ack (in this case Rqst. SN = 2). The responder recognizes the SEQ=2 and accepts the packets.

In Fig. 9, the response packet with SEQ = 1 is corrupted. The missing response is detected when the requestor times out its transaction. The requestor resets in send engine to start generating packets at the one that failed to receive an ACK (in this case Rqst. SN = 1). The responder recognizes the resent packets as already having been received, Acks them and throws the data away.

Note that the response packets for this particular RDMA transaction all have the same value of SEQ because the request SNs and response SNs are not incremented for RDMA transactions that are unordered. In this case the TSNs are utilized by requestor to match response packets to outstanding requests.

Error recovery places several requirements on the requestor's KA (Kernel Agent, the kernel mode driver code responsible for SAN error recovery):

1) The KA must determine the sequence number to restart with.

2) The KA must determine the proper data contents of the packet to be resent.

3) In order for the KA to determine the appropriate sequence number, it must be aware of how the hardware packetizes data under any given combination of descriptors, data segments, page crossings etc.

Note the responder side does not require KA involvement (unless a barrier operation fails).

The invention has now been described with reference to the preferred embodiments. Alternatives and substitutions will now be apparent to persons of skill in the art. For example, the invention has been described in the context of the ServerNet II SAN, the principles of the invention are useful in any network that utilizes multiple paths between end nodes. Accordingly, it is not intended to limit the invention except as provided by the appended claims.

WHAT IS CLAIMED IS:

1. In a system area network (SAN) including multiple nodes coupled by a network fabric, with the system for transferring data between a requestor node and a responder node, with the requestor and responder nodes coupled by first and second paths and with the SAN implementing data transfers as a sequence of request/response packet pairs, and with the SAN for implementing ordered transactions requiring that packets be received in the order transmitted and remote direct memory access packets that may be received out of order, a method for detecting and recovering from errors, said method comprising the steps of:

maintaining, at said requestor, the request out status of each of said paths as good or bad, so that only good paths are utilized for data transfer transactions;

maintaining, at said responder, a request in status of each of said paths as good or bad, so that requests are accepted only on good paths;

detecting, as said requestor, if the first path fails, and implementing a barrier transaction, to determine whether said failure is transient or permanent;

if transient, at said requestor, retrying said transaction on said first path;

if permanent, at said requestor, utilizing the second path to inform said responder that the first path is bad;

at said responder, updating the responder request in status of the first path to indicate that the first path is bad so that packets will not be accepted from said first path;

at the requestor, updating the request out status of the first path to indicate that the first path is bad so that subsequent transaction do not use said first path.

2. The method of claim 1 wherein said step of detecting comprises:

at the requestor; maintaining a request sequence number and an elapsed time since the beginning of a transaction including the request sequence number as a packet sequence number in a request packet and, for an ordered transaction, incrementing the request sequence number after the request packet is sent;

at the responder: maintaining a responder sequence number; sending an acknowledge packet for each received request packet indicating whether its packet sequence number matches the responder sequence number and, incrementing the

responder sequence number only if the packet sequence number matches the responder sequence number and the transaction is an ordered transaction; and

at the requestor: indicating that the first path is bad if, after a fixed time has elapsed, an acknowledge packet has not been received for all request packets sent indicating that the packet sequence number matched the responder sequence number.

3. The method of claim 1 wherein said step of implementing a barrier transaction in order to verify the correct operation of path and to ensure there are no remaining request or response packets in SAN, comprises the steps of:

at the requestor, implementing a remote direct access memory write operation along said first path to write an arbitrary value at said responder and subsequently implementing a remote direct memory access read operation to read the arbitrary value from the responder.

4. A system area network (SAN) including multiple nodes coupled by a network fabric, with the system for transferring data between a requestor node and a responder node, with the requestor and responder nodes coupled by first and second paths and with the SAN implementing data transfers as a sequence of request/response packet pairs, and with the SAN for implementing ordered transactions requiring that packets be received in the order transmitted and remote direct memory access packets that may be received out of order, a system for detecting and recovering from errors, said system comprising:

a requestor end node including a controller for executing kernel agent software and a requestor network interface card (NIC), forming a part of a requestor node, with the requestor NIC including requestor status logic and transmission logic, with said requestor status logic maintaining the request out status of each of said paths as good or bad, so that only good paths are utilized for data transfer transactions, and, if the first path is determined to be bad, updating the request out status of the first path to indicate that the first path is bad so that subsequent transaction do not use said first path and with the kernel agent software:

detecting, as said requestor, if the first path fails, and implementing a barrier transaction, to determine whether said failure is transient or permanent;

if transient, at said requestor, retrying said transaction on said first path;

if permanent, at said requestor, utilizing the second path to inform said responder that the first path is bad; and with the SAN comprising:

a responder end node including a responder network interface card (NIC), forming a part of a responder node, with the responder NIC including responder status logic and reception logic, with said responder status logic maintaining a request in status of each of said paths as good or bad, so that requests are accepted only on good paths, and, if the first path is determined to be bad, updating the responder request in status of the first path to indicate that the first path is bad so that packets will not be accepted from said first path.

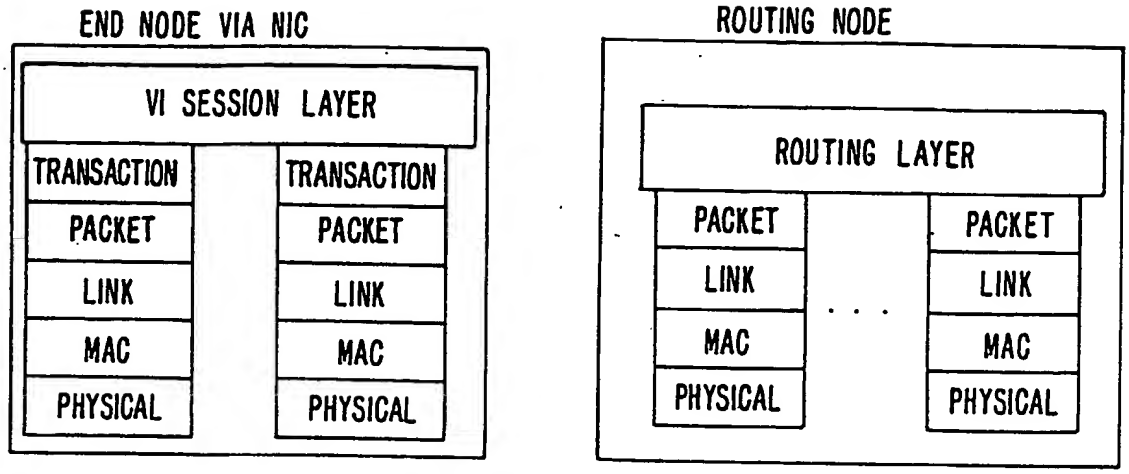

END NODE VIA NIC
VI SESSION LAYER
TRANSACTION
PACKET
LINK
MAC
PHYSICAL
TRANSACTION
PACKET
LINK
MAC
PHYSICAL
ROUTING NODE
ROUTING LAYER
PACKET
LINK
MAC
PHYSICAL
PACKET
LINK
MAC
PHYSICAL

FIG. 1.

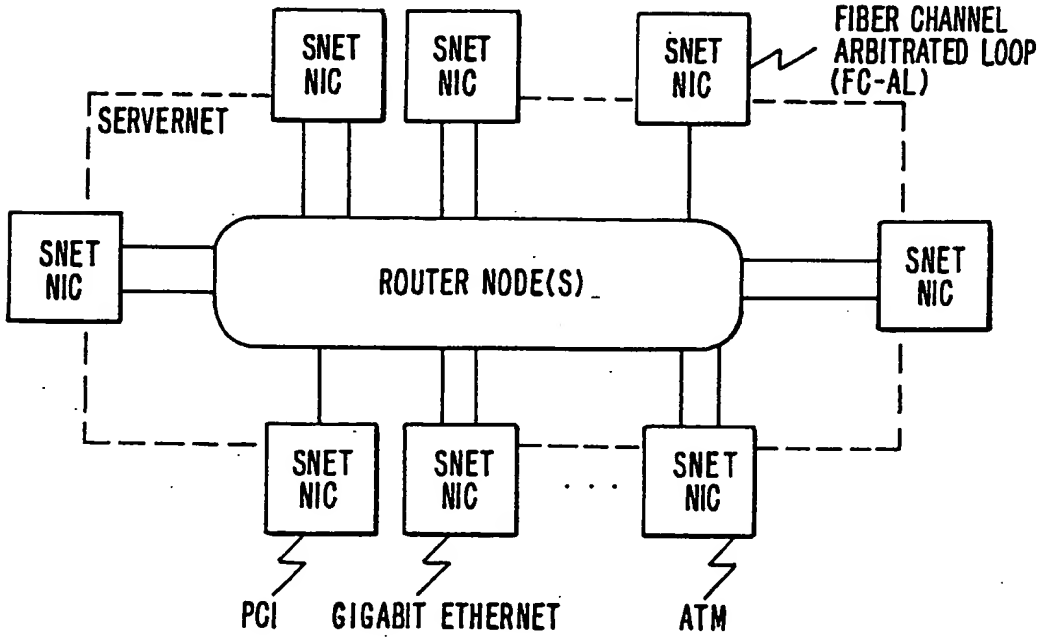

SNET NIC
SNET NIC
SNET NIC
FIBER CHANNEL ARBITRATED LOOP (FC-AL)
SERVERNET
SNET NIC
ROUTER NODE(S)
SNET NIC
SNET NIC
SNET NIC
SNET NIC
PCI
GIGABIT ETHERNET
ATM

FIG. 2.

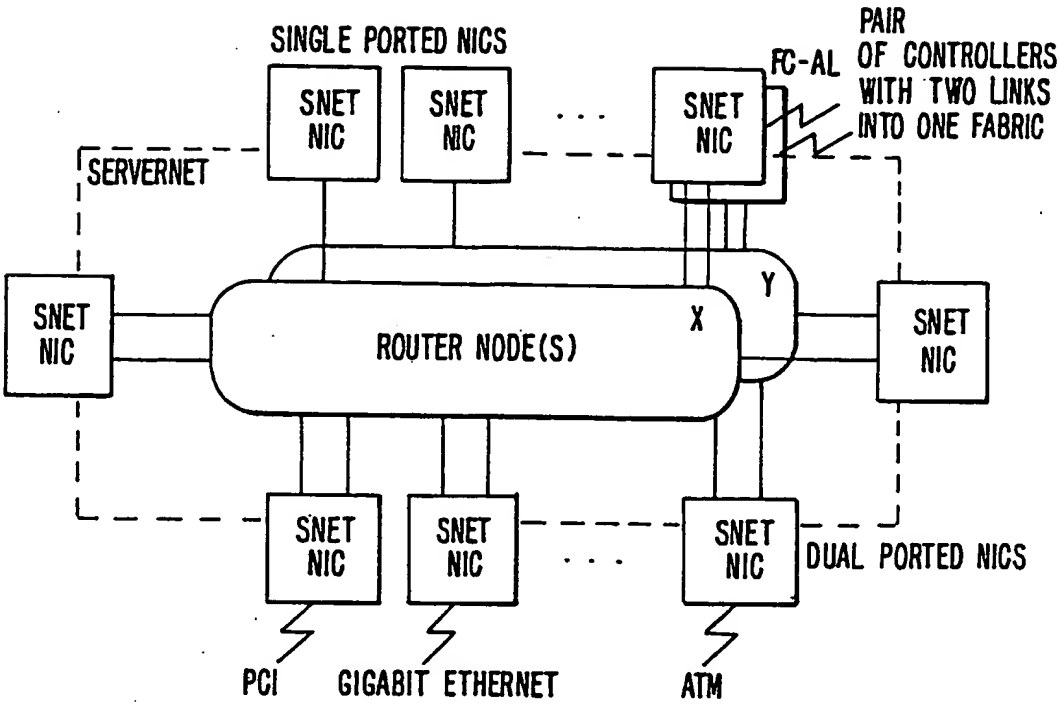

SINGLE PORTED NICS
SNET NIC
SNET NIC
SNET NIC
FC-AL
PAIR OF CONTROLLERS WITH TWO LINKS INTO ONE FABRIC
SERVERNET
SNET NIC
ROUTER NODE(S)
X
Y
SNET NIC
SNET NIC
SNET NIC
SNET NIC
DUAL PORTED NICS
PCI
GIGABIT ETHERNET
ATM

FIG. 3.

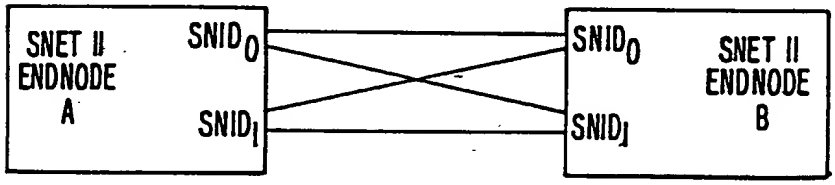

SNET II ENDNODE A
SNID0
SNID1
SNID0
SNID1
SNET II ENDNODE B

FIG. 4.

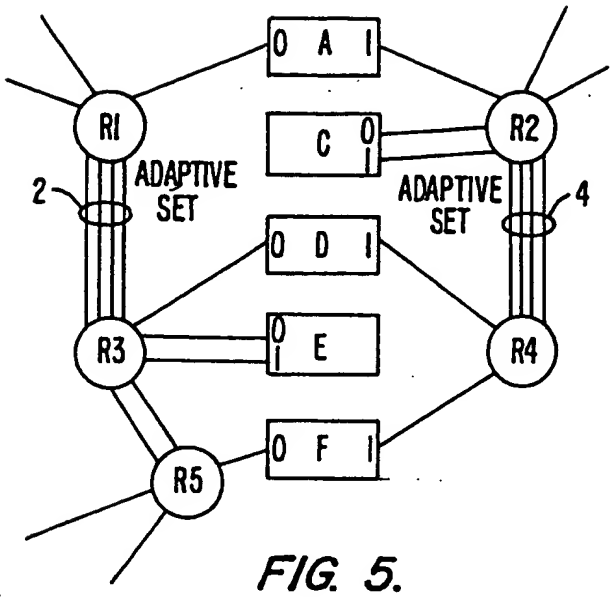

0 A 1
R1
R2
C 0 1
ADAPTIVE SET
2
ADAPTIVE SET
4
0 D 1
R3
0 1 E
R4
0 F 1
R5

FIG. 5.

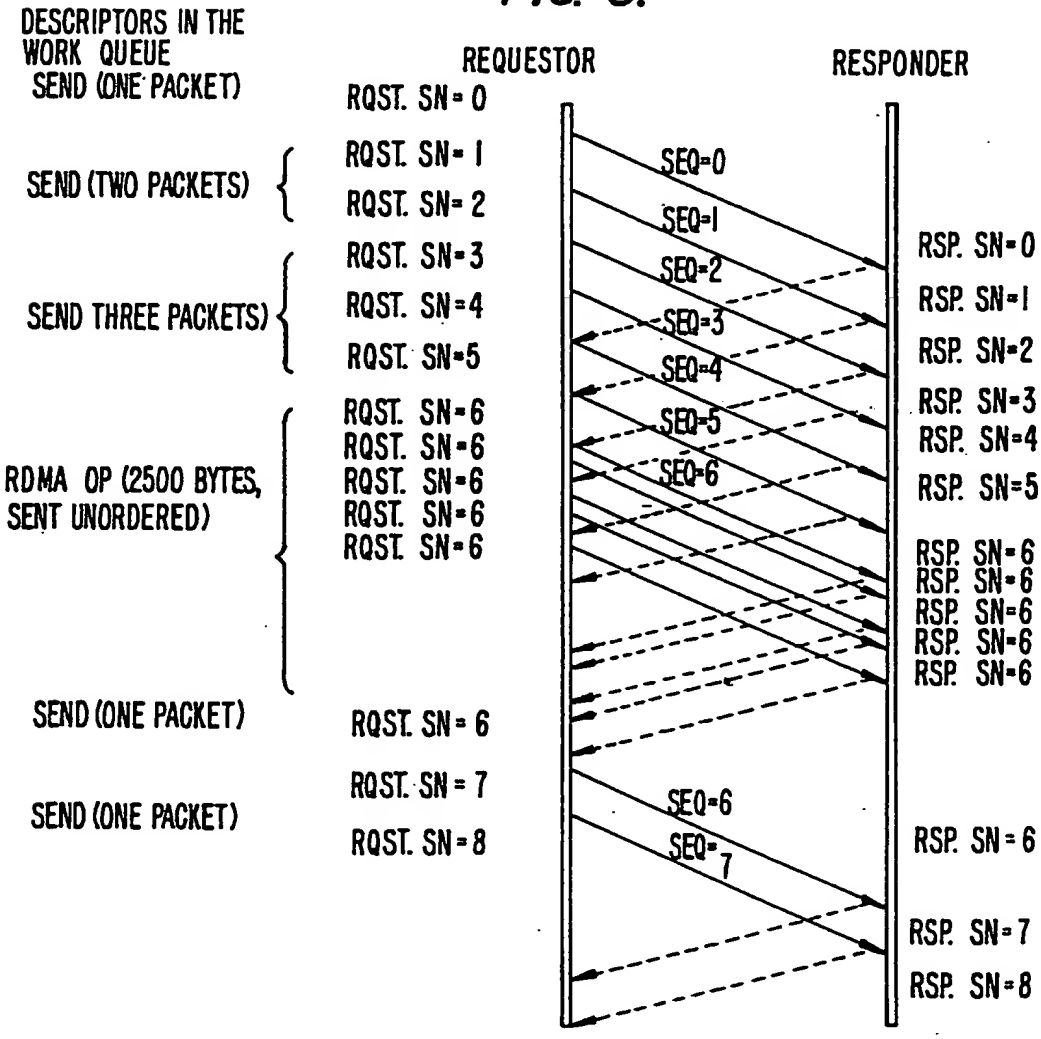

DESCRIPTORS IN THE WORK QUEUE
REQUESTOR
RESPONDER
SEND (ONE PACKET)
RQST. SN = 0
SEND (TWO PACKETS)
RQST. SN = 1
RQST. SN = 2
SEND THREE PACKETS)
RQST. SN = 3
RQST. SN = 4
RQST. SN = 5
RDMA OP (2500 BYTES, SENT UNORDERED)
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
SEND (ONE PACKET)
RQST. SN = 6
SEND (ONE PACKET)
RQST. SN = 7
RQST. SN = 8
SEQ=0
SEQ=1
SEQ=2
SEQ=3
SEQ=4
SEQ=5
SEQ=6
SEQ=6
SEQ=7
RSP. SN = 0
RSP. SN = 1
RSP. SN = 2
RSP. SN = 3
RSP. SN = 4
RSP. SN = 5
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 7
RSP. SN = 8

FIG. 6.

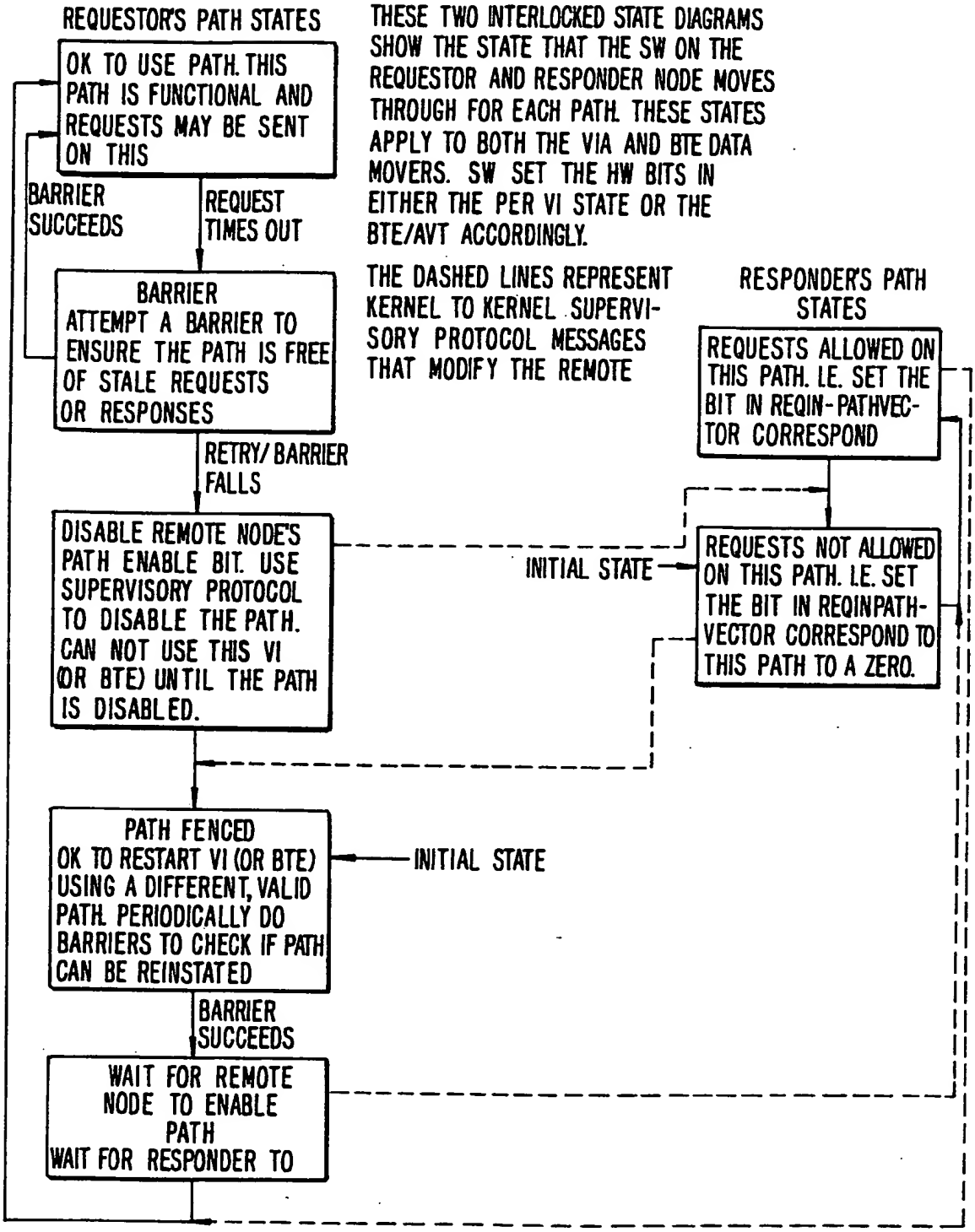

REQUESTOR'S PATH STATES
OK TO USE PATH. THIS PATH IS FUNCTIONAL AND REQUESTS MAY BE SENT ON THIS
BARRIER SUCCEEDS
REQUEST TIMES OUT
BARRIER
ATTEMPT A BARRIER TO ENSURE THE PATH IS FREE OF STALE REQUESTS OR RESPONSES
RETRY/ BARRIER FALLS
DISABLE REMOTE NODE'S PATH ENABLE BIT. USE SUPERVISORY PROTOCOL TO DISABLE THE PATH. CAN NOT USE THIS VI (OR BTE) UNTIL THE PATH IS DISABLED.
PATH FENCED
OK TO RESTART VI (OR BTE) USING A DIFFERENT, VALID PATH. PERIODICALLY DO BARRIERS TO CHECK IF PATH CAN BE REINSTATED
INITIAL STATE
BARRIER SUCCEEDS
WAIT FOR REMOTE NODE TO ENABLE PATH
WAIT FOR RESPONDER TO
THESE TWO INTERLOCKED STATE DIAGRAMS SHOW THE STATE THAT THE SW ON THE REQUESTOR AND RESPONDER NODE MOVES THROUGH FOR EACH PATH. THESE STATES APPLY TO BOTH THE VIA AND BTE DATA MOVERS. SW SET THE HW BITS IN EITHER THE PER VI STATE OR THE BTE/AVT ACCORDINGLY.
THE DASHED LINES REPRESENT KERNEL TO KERNEL SUPERVISORY PROTOCOL MESSAGES THAT MODIFY THE REMOTE
RESPONDER'S PATH STATES
REQUESTS ALLOWED ON THIS PATH. I.E. SET THE BIT IN REQIN-PATHVECTOR CORRESPOND
INITIAL STATE
REQUESTS NOT ALLOWED ON THIS PATH. I.E. SET THE BIT IN REQINPATHVECTOR CORRESPOND TO THIS PATH TO A ZERO.

FIG. 7.

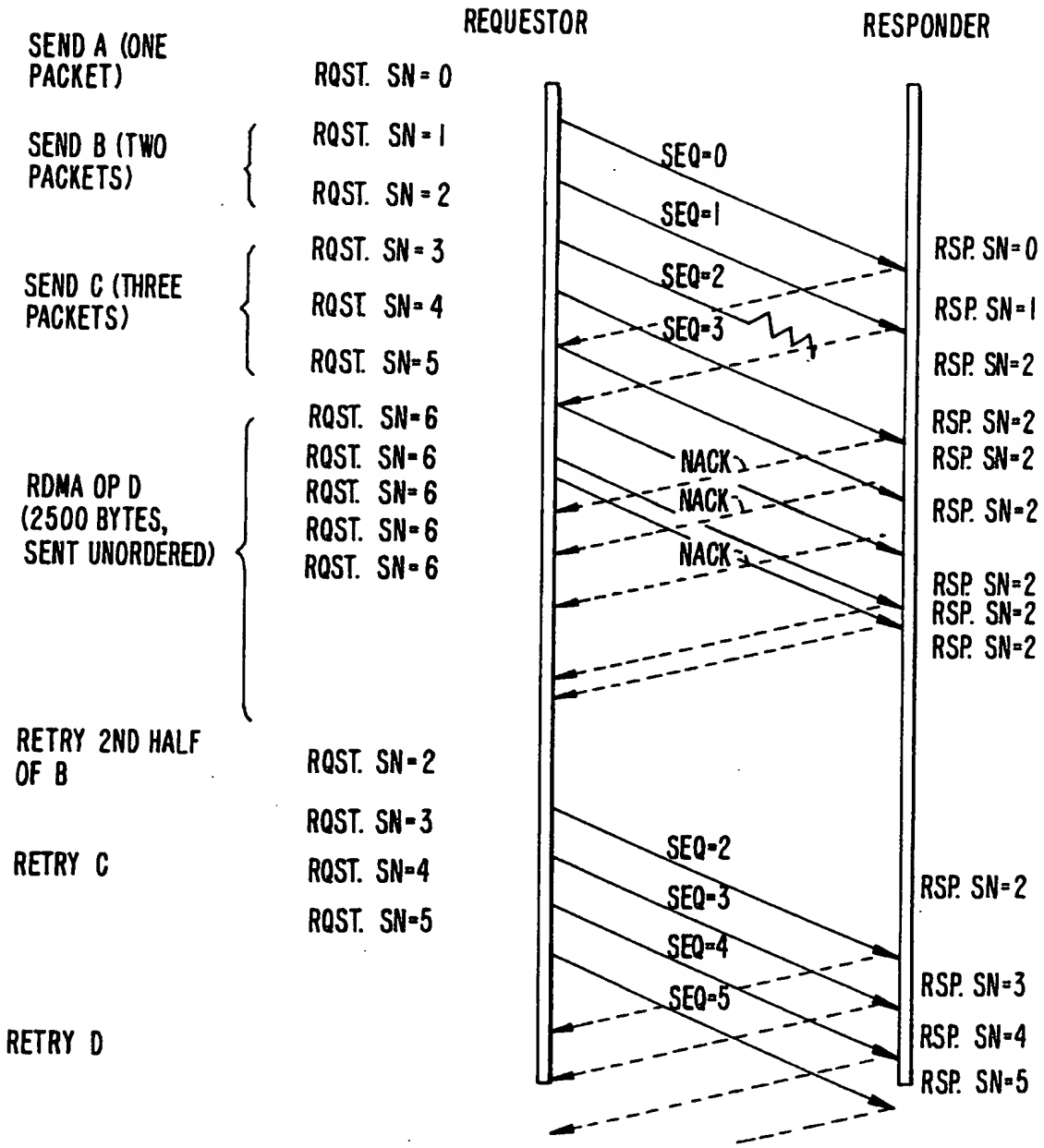

DESCRIPTORS IN THE WORK QUEUE
REQUESTOR
RESPONDER
SEND A (ONE PACKET)
RQST. SN = 0
SEND B (TWO PACKETS)
RQST. SN = 1
RQST. SN = 2
SEND C (THREE PACKETS)
RQST. SN = 3
RQST. SN = 4
RQST. SN = 5
RDMA OP D (2500 BYTES, SENT UNORDERED)
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RETRY 2ND HALF OF B
RQST. SN = 2
RQST. SN = 3
RETRY C
RQST. SN = 4
RQST. SN = 5
RETRY D
SEQ=0
SEQ=1
SEQ=2
SEQ=3
NACK
NACK
NACK
SEQ=2
SEQ=3
SEQ=4
SEQ=5
RSP. SN=0
RSP. SN=1
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=2
RSP. SN=3
RSP. SN=4
RSP. SN=5

FIG. 8.

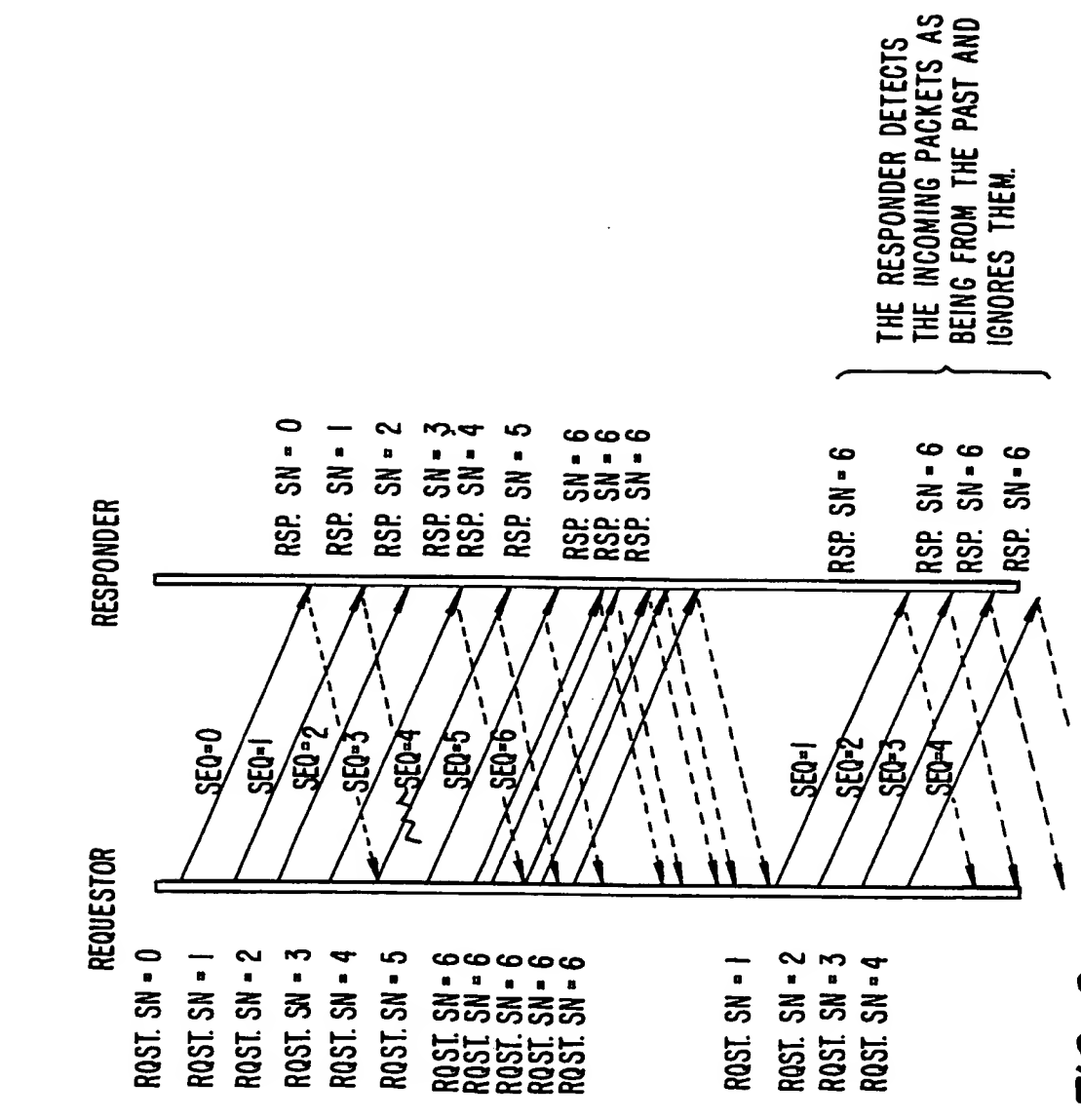

DESCRIPTORS IN THE WORK QUEUE
SEND A (ONE PACKET)
SEND B (TWO PACKETS)
SEND C (THREE PACKETS)
RDMA OP D (2500 BYTES, SENT UNORDERED)
RETRY B
RETRY C
RETRY D
REQUESTOR
RESPONDER
RQST. SN = 0
RQST. SN = 1
RQST. SN = 2
RQST. SN = 3
RQST. SN = 4
RQST. SN = 5
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 6
RQST. SN = 1
RQST. SN = 2
RQST. SN = 3
RQST. SN = 4
SEQ=0
SEQ=1
SEQ=2
SEQ=3
SEQ=4
SEQ=5
SEQ=6
SEQ=1
SEQ=2
SEQ=3
SEQ=4
RSP. SN = 0
RSP. SN = 1
RSP. SN = 2
RSP. SN = 3
RSP. SN = 4
RSP. SN = 5
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
RSP. SN = 6
THE RESPONDER DETECTS THE INCOMING PACKETS AS BEING FROM THE PAST AND IGNORES THEM.

FIG. 9.

INTERNATIONAL SEARCH REPORT

Inter ɔnal Application No
PCT/US 99/00249

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L12/56 H04L12/44 H04L29/14 H04L12/26 H04L29/06 H04L1/18

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H01L H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 757 318 A (TANDEM COMPUTERS INCORPORATED) 5 February 1997 see page 51, line 40 - page 53, line 28; table 7 see page 60, line 55 - page 62, line 40	1-4
A	D. GARCIA ET AL.: "ServerNet II" PARALLEL COMPUTER ROUTING AND COMMUNICATION (2ND INT. WKSP), 26 June 1997, pages 119-135, XP002103164 Atlanta, USA	1-4

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 May 1999	07/06/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Absalom, R

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter ional Application No
PCT/US 99/00249

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 757318 A	05-02-1997	CA 2178393 A	08-12-1996
		JP 9134337 A	20-05-1997

```
Set     Items   Description
S1   10701879   MAP???? OR TRAC??? OR PLAN???? OR PROJECT??? OR
             STRATEG??? OR PREPLAN????
S2    8891017   FLOW??? OR PROGRESS??? OR CONTINU????? OR COURSE? ? OR
             MOVEMENT OR SEQUENCE OR SUCCESSION
S3   10855653   DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR KB??
S4    4419400   THREAD? ? OR CONNECT??? ? OR LINK? ? OR ATTACH???? OR
             PATH? ? OR CIRCUIT? ?
S5   10387594   INIT?????? OR SOURCE OR START??? OR BEGIN???? OR
             ORIGIN?? OR FIRST
S6    7038321   NODE? ? OR FUNCTIONAL()BLOCK? ? OR  COMPONENT? ? OR
             CONSTITUENT? ? OR ELEMENT? ?
S7    4725989   FINAL OR TARGET? ? OR TERMINAL? ? OR DESTINATION? ? OR
             END??? OR LAST
S8     210992   BUSY OR OCCUP??? OR UNAVAIL????
S9    8512344   SERVICE? ? OR QOS OR FUNCTION? ? OR TRANSACTION? ? OR
             JOB? ? OR TASK? ?
S10   1986195   GUARANTEE? ? OR AGREEMENT? ? OR CONTRACT? ? OR GUARANTY
S11     87229   ASIC OR DRAM OF FPGA OR VLSI OR SYSTEM(3N)(CHIP? ? OR
             CIRCUIT? ?)
S12         0   S1 AND S2 AND S3 AND S4 AND S5 AND S6 AND S7 AND S8 AND
             S9 AND S10 AND S11
S13         1   S1 AND S2 AND S3 AND S4 AND S5 AND S6 AND S7 AND S8 AND
             S9 AND S10
S14         7   (S4 AND S5 AND S6 AND S7 AND S8 AND S9 AND S10)
             NOT S13
? show files
File   2:INSPEC 1898-2006/Jul W3
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Jul W3
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Jul W3
         (c) 2006 Elsevier Eng.  Info. Inc.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Jul W4
         (c) 2006 The Thomson Corp
File  35:Dissertation Abs Online 1861-2006/Jun
         (c) 2006 ProQuest Info&Learning
File  56:Computer and Information Systems Abstracts 1966-2006/Jul
         (c) 2006 CSA.
File  57:Electronics & Communications Abstracts 1966-2006/Jul
         (c) 2006 CSA.
File  60:ANTE: Abstracts in New Tech & Engineer 1966-2006/Jul
         (c) 2006 CSA.
File  65:Inside Conferences 1993-2006/Jul 27
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Apr W4
         (c)2006 Japan Science and Tech Corp(JST)
File  95:TEME-Technology & Management 1989-2006/Jul W4
         (c) 2006 FIZ TECHNIK
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/Jul
         (c) 2006 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2006/Jul 17
         (c) 2006 The Gale Group
File 144:Pascal 1973-2006/Jul W1
         (c) 2006 INIST/CNRS
File 256:TecInfoSource 82-2006/Oct
         (c) 2006 Info.Sources Inc
```

File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec

12/9/6 (Item 3 from file: 35)
DIALOG(R)File 35:Dissertation Abs Online

01431601 ORDER NO: AADAA-I9529456
THE DYNAMIC MANAGEMENT OF GUARANTEED -PERFORMANCE CONNECTIONS IN PACKET SWITCHED INTEGRATED- SERVICES NETWORKS
Author: PARRIS, COLIN JAMES
Degree: PH.D.
Year: 1994
Corporate Source/Institution: UNIVERSITY OF CALIFORNIA, BERKELEY (0028)
Chair: DOMENICO FERRARI
Source: VOLUME 56/05-B OF DISSERTATION ABSTRACTS INTERNATIONAL.
PAGE 2727. 161 PAGES
Descriptors: COMPUTER SCIENCE
Descriptor Codes: 0984

In this thesis, we examine this problem of flexibility of **Guaranteed** Performance Communication (GPC) **services** in wide-area packet-switched networks, and present a solution by proof of concept; that is, we designed a dynamic resource management scheme, analyzed its behavior through simulation experiments, and implemented a prototype of the scheme. This dynamic resource management scheme, called the Dynamic **Connection** Management (DCM) scheme, provides the network with the capability to dynamically modify the traffic characteristics, the performance requirements, and the routes of any existing **guaranteed** -performance **connection** . We **begin** this examination by providing several examples of the dynamics of the client demands and of the network state to motivate our work, and we continue with a review and critique of various proposed solutions. We then present the concept of Dynamic **Connection** Management and discuss its **components** : namely, the DCM scheme and the DCM Policies. The DCM scheme is a collection of algorithms and mechanisms that permit the runtime modification of the traffic and performance parameters, and the route of a **guaranteed** -performance **connection** . The DCM scheme is guided by high-level management policies, called the DCM policies, that determine when modification is permissible in the network and the values of the appropriate parameters to be modified. The focus of this thesis is the DCM scheme.

The DCM scheme is an enhancement of the Tenet GPC **service** ; it is based on three algorithms: the DCM channel administration algorithm, the DCM transition algorithm, and the DCM routing algorithm; and it is subject to the DCM modification **contract** . This **contract** specifies the degree of disruption that a client may experience during a modification. This degree can range from no disruption to a bounded number of performance violations. The channel administration algorithm conducts the admission control tests and reserves the appropriate network resources to ensure that the performance **guarantees** of the modified channel are satisfied during and after modification. The DCM transition algorithm ensures that the performance violations specified in the DCM modification **contract** are adhered to during modification. The DCM routing algorithm determines a route from the **source** to the **destination** host according to the traffic and performance requirements and other administrative factors.

The DCM scheme also supports mechanisms that enable modifications to a **connection** to be made to a segment of the **connection** (local control) or to the entire **connection** (global control). Furthermore, faster establishment and modification is also possible as the DCM scheme uses the intelligent restart establishment mechanism, which utilizes the real-time network and client state to compute the **path** before establishment and the time value of the network state information to bypass **unavailable links** during establishment. The DCM scheme was verified and analyzed by a series

of simulation experiments. These simulation experiments indicated that the scheme is functionally correct and that the performance of the scheme is very acceptable given our time scales of interest. (Abstract shortened by UMI.)
?

11/9/1 (Item 1 from file: 34)
DIALOG(R)File 34:SciSearch(R) Cited Ref Sci

02809635 Genuine Article#: MF633 Number of References: 38
Title: AN ANALYTICAL MODEL FOR ADAPTIVE ROUTING NETWORKS
Author(s): ASH GR; HUANG BSD
Corporate Source: AT&T BELL LABS/HOLMDEL//NJ/07733
Journal: IEEE TRANSACTIONS ON COMMUNICATIONS, 1993, V41, N11 (NOV), P 1748-1759
ISSN: 0090-6778
Language: ENGLISH Document Type: ARTICLE
Geographic Location: USA
Subfile: SciSearch; CC ENGI--Current Contents, Engineering, Technology & Applied Sciences
Journal Subject Category: TELECOMMUNICATIONS; ENGINEERING, ELECTRICAL & ELECTRONIC
Abstract: Real-time network routing (RTNR) is a new adaptive routing method which replaced dynamic nonhierarchical routing (DNHR) in the AT&T network **starting** in 1991. RTNR is being introduced to extend dynamic routing to all new and existing **services** , and to increase network robustness. With RTNR, switches have a simple way of exchanging **link** status **bit map information** , thereby determining the availability and load conditions of the direct and all two- **link paths** to the **destination** . **Link busy** -idle status is exchanged between the network **nodes** using a **bit map data** exchange through the common channel signaling (CCS) network, and calls are set up where there is the most available capacity in the network To date the analysis of RTNR networks has been limited to simulation models, in part because of the lack of analytical models for such networks. In this paper, an analytical model is developed for the AT&T network under RTNR, and is shown to provide good **agreement** with simulation models.

The analytical model for RTNR networks uses an erlang fixed point method to solve the nonlinear equations describing dynamical network behavior. The equations include the **link** state probability, network **flows** , **link** arrival rates, adaptive trunk reservation level, and adaptive **path** selection depth. The **link** state model provides the aggregate **link** state probabilities through solution of the birth-death equations, and models the adaptive nature of trunk reservation. The network **flow** model provides a method to calculate the traffic **flow** using the least **busy** concept employed in RTNR, and also models the adaptive nature of the **path** selection depth. The analytical model addresses asymmetrical networks, and computational examples show the differences from the simulation model to be small. We also use the analytical model to examine key RTNR parameters over a range of values, and the model provides validation of some of the parameter values selected for **initial** RTNR implementation.
Cited References:
TELESIS MAG, 1986
AKINPELU JM, 1984, V63, BELL SYST TECH J
ASH GR, 1981, V60, BELL SYST TECH J
ASH GR, 1981, V60, BELL SYST TECH J
ASH GR, 1989, V7, IEEE J SELECT AREAS
ASH GR, 1989, NOV P IEEE GLOB TEL
ASH GR, 1989, SEP NETW MAN CONTR W
ASH GR, 1983, 10TH P INT TEL C MON
ASH GR, 1985, 11TH P INT TEL C KYO
ASH GR, 1988, 12TH P INT TEL C TOR
ASH GR, 1991, 13TH P INT TEL C COP

CAMERON WH, 1980, P NETWORKS C PARIS
CARON F, 1988, 12TH P INT TEL C TOR
CARROLL JJ, 1983, DEC P IEEE GLOB TEL
CHEMOUIL P, 1986, V11, COMPUT NETWORKS ISDN
CHUNG SP, 1990, REDUCED LOAD APPROXI
FIELD FA, 1983, 10TH P INT TEL C MON
GARCIA JM, 1985, 11TH P INT TEL C KYO
GAUTHIER P, 1987, P IEEE GLOBAL TELECO
GIBBENS RJ, 1986, STATIST LAB JAN
HURLEY BR, 1987, V25, IEEE COMMUN MAG
KATZ SS, 1988, SEP INT SEM TEL NETW
KATZ SS, 1967, 5TH P INT TEL C NY
KELLY FP, 1986, V18, ADV APPLIED PROBABIL
KRISHNAN KR, 1988, 25TH P C DEC CONTR A
LANGLOIS F, 1991, 13TH P INT TEL C COP
MEES A, 1986, V323, P108, NATURE
MITRA D, 1991, IEEE T COMM JAN
MITRA D, 1991, 13TH P INT TEL C COP
NARENDRA KS, 1977, V7, IEEE T SYST MAN CYBE
NARENDRA KS, 1980, V10, IEEE T SYST MAN CYBE
REGNIER J, 1983, 10TH P INT TEL C MON
STACEY RR, 1987, MAR P INT SWITCH S P
WANAMAKER DM, 1988, MAR P NETW OP MAN SY
WATANABE Y, 1987, 5TH P ITC SPEC SEM T
WHITT W, 1985, V64, AT T TECH J
WONG EWM, 1990, P IEEE INFOCOM 90
WONG EWM, SELECTIVE ALTERNATE
?

```
Set     Items   Description
S1    2444793   MAP???? OR TRAC??? OR PLAN???? OR PROJECT??? OR STRATEG???
            OR PREPLAN????
S2    3802635   FLOW??? OR PROGRESS??? OR CONTINU????? OR COURSE? ? OR MOV-
            EMENT OR SEQUENCE OR SUCCESSION
S3    3386609   DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR KB??
S4    7517179   THREAD? ? OR CONNECT??? ? OR LINK? ? OR ATTACH???? OR PATH?
             ? OR CIRCUIT? ?
S5    5000862   INIT?????? OR SOURCE OR START??? OR BEGIN???? OR ORIGIN?? -
            OR FIRST
S6    4709211   NODE? ? OR FUNCTIONAL()BLOCK? ? OR  COMPONENT? ? OR CONSTI-
            TUENT? ? OR ELEMENT? ?
S7    4827836   FINAL OR TARGET? ? OR TERMINAL? ? OR DESTINATION? ? OR END-
            ??? OR LAST
S8     108903   BUSY OR OCCUP??? OR UNAVAIL????
S9    1371263   SERVICE? ? OR QOS OR FUNCTION? ? OR TRANSACTION? ? OR JOB?
            ? OR TASK? ?
S10     91560   GUARANTEE? ? OR AGREEMENT? ? OR CONTRACT? ? OR GUARANTY
S11    121797   ASIC OR DRAM OR FPGA OR VLSI OR SYSTEM(3N)(CHIP? ? OR CIRC-
            UIT? ?)
S12         0   S1 AND S2 AND S3 AND S4 AND S5 AND S6 AND S7 AND S8 AND S9
             AND S10 AND S11
S13         0   S1 AND S2 AND S3 AND S4 AND S5 AND S6 AND S7 AND S8 AND S9
             AND S10
S14         4   S4 AND S5 AND S6 AND S7 AND S8 AND S9 AND S10
S15         9   (S1 AND S3 AND S4 AND S8 AND S9 AND S10) NOT S14
S16     14851   S9 AND S10
S17       143   S16 AND S11
S18         0   (S17 AND IC=(G06F-009/46 OR G06F-015/163 OR G06F-009/54 OR
             G06F-009/00)) NOT (S14:S15 OR AD=(20000309:20030309) OR AD=(2-
            0030309:20060728))
S19       111   (S16 AND IC=(G06F-009/46 OR G06F-015/163 OR G06F-009/54 OR
             G06F-009/00)) NOT (S14:S15 OR AD=(20000309:20030309) OR AD=(2-
            0030309:20060728))
S20        54   S19 AND (S9 OR S10)/TI
S21         1   AU=((WEBER W? OR WEBER, W?) AND (ARAS R? OR ARAS, R?) AND -
            (ROBINSON L? OR ROBINSON, L?) AND (ROSSEEL G? OR ROSSEEL G?) -
            AND (TOMLINSON J? OR TOMLINSON, J) AND (WINGARD D? OR WINGARD,
             D?))
S22         1   (AU=(WEBER W? OR WEBER, W? OR ARAS R? OR ARAS, R? OR ROBIN-
            SON L? OR ROBINSON, L? OR ROSSEEL G? OR ROSSEEL G? OR TOMLINS-
            ON J? OR TOMLINSON, J OR WINGARD D? OR WINGARD, D?) AND BUSY)
            NOT (S14:S15 OR S20 OR AD=(20000309:20030309) OR AD=(20030309-
            :20060728))
? show files
File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
        (c) 2006 JPO & JAPIO
File 350:Derwent WPIX 1963-2006/UD=200647
        (c) 2006 The Thomson Corporation
?
```

21/5/1 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0013541084 - Drawing available
WPI ACC NO: 2003-634750/
XRPX Acc No: N2003-504802
Data communication method for electronic computing system, involves stopping issue of data transfer between initiator block and functional block in response to issued busy signal
Patent Assignee: ARAS R (ARAS-I); ROBINSON L A (ROBI-I); ROSSEEL G P (ROSS-I); SONICS INC (SONI-N); TOMLINSON J S (TOML-I); WEBER W (WEBE-I); WINGARD D E (WING-I)
Inventor: **ARAS R ; ROBINSON L A ; ROSSEEL G P ; TOMLINSON J S ; WEBER W ; WINGARD D E**
Patent Family (5 patents, 99 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	
US 20020129173	A1	20020912	US 2001802405	A	20010309	200360	B
WO 2002073407	A1	20020919	WO 2002US5015	A	20020219	200360	E
EP 1370939	A1	20031217	EP 2002709610	A	20020219	200402	E
			WO 2002US5015	A	20020219		
AU 2002244087	A1	20020924	AU 2002244087	A	20020219	200433	E
JP 2004530197	W	20040930	JP 2002571999	A	20020219	200465	E
			WO 2002US5015	A	20020219		

Priority Applications (no., kind, date): US 2001802405 A 20010309

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
US 20020129173	A1	EN	31	21	
WO 2002073407	A1	EN			

National Designated States,Original: AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ OM PH PL PT RO RU SD SE SG SI SK SL TJ TM TN TR TT TZ UA UG UZ VN YU ZA ZM ZW
Regional Designated States,Original: AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW MZ NL OA PT SD SE SL SZ TR TZ UG ZM ZW

EP 1370939	A1	EN		PCT Application WO 2002US5015 Based on OPI patent WO 2002073407

Regional Designated States,Original: AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI TR

AU 2002244087	A1	EN		Based on OPI patent WO 2002073407
JP 2004530197	W	JA	96	PCT Application WO 2002US5015 Based on OPI patent WO 2002073407

Alerting Abstract US A1

NOVELTY - A busy signal identified by a thread identifier, associating a data transfer with a transaction stream, is issued if a target block is unable to accept the data from an initiator block (816). Issue of data transfer is withheld by the initiator block, in response to the issued signal. Data transfers not associated with identifier identified by the issued signal, are issued.

DESCRIPTION - An INDEPENDENT CLAIM is also included for communication apparatus.

USE - For computer system, electronic computing and communication system.

ADVANTAGE - The data transfer at different levels can be identified without additional knowledge.

DESCRIPTION OF DRAWINGS - The figure shows the block diagram of the initiator interface module connected to shared communication bus and

initiator functional block.
800 initiator interface module
804 arbitrator block
812 synchronizer
814 shared communications bus
816 initiator functional block

Title Terms/Index Terms/Additional Words: DATA; COMMUNICATE; METHOD; ELECTRONIC; COMPUTATION; SYSTEM; STOP; ISSUE; TRANSFER; INITIATE; BLOCK; FUNCTION; RESPOND; BUSY; SIGNAL

Class Codes
International Classification (Main): G06F-013/42, G06F-009/46, G06F-009/54
(Additional/Secondary): G06F-013/38, G06F-015/163, G06F-009/00
US Classification, Issued: 709310000, 709107000

File Segment: EPI;
DWPI Class: T01
Manual Codes (EPI/S-X): T01-F02; T01-H05B3
?

```
Set     Items   Description
S1   18753100   FLOW??? OR PROGRESS??? OR CONTINU????? OR COURSE? ? OR MOV-
            EMENT OR SEQUENCE OR SUCCESSION
S2     600289   S1(3N)(DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR KB??)
S3       9467   S2(5N)(THREAD? ? OR CONNECT??? ? OR LINK? ? OR ATTACH???? -
            OR PATH? ? OR CIRCUIT? ?)
S4     168452   (INIT?????? OR SOURCE OR START??? OR BEGIN???? OR ORIGIN??
            OR FIRST)(3N)(NODE? ? OR FUNCTIONAL()BLOCK? ? OR  COMPONENT? ?
             OR CONSTITUENT? ? OR ELEMENT? ?)
S5           3   S3(5N)S4
S6       91562   (FINAL OR TARGET? ? OR TERMINAL? ? OR DESTINATION? ? OR EN-
            D??? OR LAST)(3N)(NODE? ? OR FUNCTIONAL()BLOCK? ? OR  COMPONE-
            NT? ? OR CONSTITUENT? ? OR ELEMENT? ?)
S7           0   S5(5N)S6
S8           4   S3(10N)S4
S9           0   S8(10N)S6
S10  29225715   MAP???? OR TRAC??? OR PLAN???? OR PROJECT??? OR STRATEG???
            OR PREPLAN????
S11   1394235   BUSY OR OCCUP??? OR UNAVAIL????
S12   1318766   (SERVICE? ? OR QOS OR FUNCTION? ? OR TRANSACTION? ? OR JOB?
             ? OR TASK? ?)(5N)(GUARANTEE? ? OR AGREEMENT? ? OR CONTRACT? ?
             OR GUARANTY)
S13    418430   ASIC OR DRAM OR FPGA OR VLSI OR SYSTEM(3N)(CHIP? ? OR CIRC-
            UIT? ?)
S14          0   S3 AND S4 AND S6 AND S10 AND S11 AND S12 AND S13
S15          3   S3 AND S4 AND S6 AND S10 AND S11 AND S12
S16    731318   (DATA OR INFORMATION OR BIT? ? OR BYTE? ? OR KB??)(5N)(THR-
            EAD? ? OR CONNECT??? ? OR LINK? ? OR ATTACH???? OR PATH? ? OR
            CIRCUIT? ?)
S17          6   (S1 AND S16 AND S4 AND S6 AND S10 AND S11 AND S12 AND S13)
             NOT S15
S18         34   (S1 AND S16 AND S4 AND S6 AND S10 AND S11 AND S12) NOT (S15
             OR S17)
S19         23   RD  (unique items)
S20         13   S19 AND (PY<2001 OR PD<20000309)
S21          0   AU=((WEBER W? OR WEBER, W?) AND (ARAS R? OR ARAS, R?) AND -
            (ROBINSON L? OR ROBINSON, L?) AND (ROSSEEL G? OR ROSSEEL G?) -
            AND (TOMLINSON J? OR TOMLINSON, J) AND (WINGARD D? OR WINGARD,
             D?))
S22         36   AU=(WEBER W? OR WEBER, W? OR ARAS R? OR ARAS, R? OR ROBINS-
            ON L? OR ROBINSON, L? OR ROSSEEL G? OR ROSSEEL G? OR TOMLINSON
             J? OR TOMLINSON, J OR WINGARD D? OR WINGARD, D?) AND BUSY
S23         25   RD  (unique items)
S24         16   S23 AND (PY<2001 OR PD<20000309)
? show files
File 275:Gale Group Computer DB(TM) 1983-2006/Jul 28
         (c) 2006 The Gale Group
File  47:Gale Group Magazine DB(TM) 1959-2006/Jul 28
         (c) 2006 The Gale group
File  16:Gale Group PROMT(R) 1990-2006/Jul 28
         (c) 2006 The Gale Group
File 624:McGraw-Hill Publications 1985-2006/Jul 28
         (c) 2006 McGraw-Hill Co. Inc
File 484:Periodical Abs Plustext 1986-2006/Jul W4
         (c) 2006 ProQuest
File 613:PR Newswire 1999-2006/Jul 29
         (c) 2006 PR Newswire Association Inc
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File 239:Mathsci 1940-2006/Sep
         (c) 2006 American Mathematical Society
```

File 370:Science 1996-1999/Jul W3
(c) 1999 AAAS
File 696:DIALOG Telecom. Newsletters 1995-2006/Jul 28
(c) 2006 Dialog
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Jul 28
(c) 2006 The Gale Group
File 674:Computer News Fulltext 1989-2006/Jul W3
(c) 2006 IDG Communications
File 88:Gale Group Business A.R.T.S. 1976-2006/Jul 19
(c) 2006 The Gale Group
File 369:New Scientist 1994-2006/Jul W1
(c) 2006 Reed Business Information Ltd.
File 160:Gale Group PROMT(R) 1972-1989
(c) 1999 The Gale Group
File 635:Business Dateline(R) 1985-2006/Jul 28
(c) 2006 ProQuest Info&Learning
File 15:ABI/Inform(R) 1971-2006/Jul 28
(c) 2006 ProQuest Info&Learning
File 9:Business & Industry(R) Jul/1994-2006/Jul 28
(c) 2006 The Gale Group
File 13:BAMP 2006/Jul W4
(c) 2006 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
(c) 1999 Business Wire
File 610:Business Wire 1999-2006/Jul 28
(c) 2006 Business Wire.
File 647:CMP Computer Fulltext 1988-2006/Aug W3
(c) 2006 CMP Media, LLC
File 98:General Sci Abs 1984-2005/Jan
(c) 2006 The HW Wilson Co.
File 148:Gale Group Trade & Industry DB 1976-2006/Jul 28
(c)2006 The Gale Group
File 634:San Jose Mercury Jun 1985-2006/Jul 28
(c) 2006 San Jose Mercury News
File 636:Gale Group Newsletter DB(TM) 1987-2006/Jul 28
(c) 2006 The Gale Group
?

```
File 370:Science 1996-1999/Jul W3
         (c) 1999 AAAS
File 696:DIALOG Telecom. Newsletters 1995-2006/Jul 28
         (c) 2006 Dialog
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Jul 28
         (c) 2006 The Gale Group
File 674:Computer News Fulltext 1989-2006/Jul W3
         (c) 2006 IDG Communications
File  88:Gale Group Business A.R.T.S. 1976-2006/Jul 19
         (c) 2006 The Gale Group
File 369:New Scientist 1994-2006/Jul W1
         (c) 2006 Reed Business Information Ltd.
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 635:Business Dateline(R) 1985-2006/Jul 28
         (c) 2006 ProQuest Info&Learning
File  15:ABI/Inform(R) 1971-2006/Jul 28
         (c) 2006 ProQuest Info&Learning
File   9:Business & Industry(R) Jul/1994-2006/Jul 28
         (c) 2006  The Gale Group
File  13:BAMP 2006/Jul W4
         (c) 2006  The Gale Group
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 610:Business Wire 1999-2006/Jul 28
         (c) 2006 Business Wire.
File 647:CMP  Computer Fulltext 1988-2006/Aug W3
         (c) 2006 CMP Media, LLC
File  98:General Sci Abs 1984-2005/Jan
         (c) 2006 The HW Wilson Co.
File 148:Gale Group Trade & Industry DB 1976-2006/Jul 28
         (c)2006 The Gale Group
File 634:San Jose Mercury  Jun 1985-2006/Jul 28
         (c) 2006 San Jose Mercury News
File 636:Gale Group Newsletter DB(TM) 1987-2006/Jul 28
         (c) 2006 The Gale Group
?
```